

FACULTY OF ENGINEERING, BUILT ENVIRONMENT AND INFORMATION TECHNOLOGY

FAKULTEIT INGENIEURSWESE, BOU-OMGEWING EN INLIGTINGTEGNOLOGIE



UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

An Agile and Ontology-Aided Approach for Domain-Specific Adaptations of Modelling Languages

By

Emanuele Laurenzi

BSc Information Technology (University of Camerino)

MSc Computer Science (University of Camerino)

MSc Business Information Systems (University of Applied Sciences Northwestern Switzerland)

Submitted in partial fulfilment of the requirements for the degree

Philosophiae Doctor in Information Systems

in the

FACULTY OF INFORMATICS

at the

UNIVERSITY OF PRETORIA

Supervisor

Prof. Dr. Knut Hinkelmann

Co-supervisors

Prof. Dr. Alta van der Merwe

Prof. Dr. Ulrich Reimer

ACKNOWLEDGEMENT

Undertaking this PhD has been an incredible (research) journey during which I have had the good fortune to meet and work with many great people.

It is nearly impossible to put into words how huge my gratitude is towards my main supervisor Prof. Dr. Knut Hinkelmann. Knut accepted me as a PhD student at the time when another door had suddenly closed. In retrospect, I can tell you that the closed door was for the better. It allowed a far more valuable door to open. Thanks to Knut I have not only increased my scientific knowledge but have also developed as a person. Discussions with him have always been highly insightful, enthusiastic and engaging, and from which I have always had something new to learn. Knut taught me a lifestyle with the motto “*Love what you do and do what you love*” by applying it daily. Despite his extremely busy schedule, he has been constantly available to provide meaningful guidance and detailed feedback, without which, this PhD would not have been achievable. I will be forever grateful to Knut.

I would also like to say a big thank you to my co-supervisor Prof. Dr. Alta van der Merwe, who throughout these past 4 years has always been very supportive. Alta’s energy, kindness and immense knowledge is what convinced me to pursue a PhD from the University of Pretoria (UP). During the initial preparatory period for the PhD, Alta welcomed me like a family member in her house in Pretoria. She placed me in the best situations to increase my knowledge of research methodologies and to get a solid start with my PhD. Thanks to Alta I also learned a new culture, gained a second family in South Africa and met outstanding researchers at UP.

I greatly appreciate the support received from my co-supervisor Prof. Dr. Ulrich Reimer, who introduced me to the research field of Domain-Specific Modelling Languages. I thank Ulrich very much for encouraging me to pursue a PhD and for supporting the initial development of my scientific profile. His insightful comments and challenging questions provided me with the incentive to widen my research from different perspectives.

I would also like to thank Rhona van der Merwe, who has always reacted promptly and kindly to any administrative concerns there have been.

A special thanks goes to both my current and former colleagues for showing constant and priceless support, also for asking: *so, when are you going to finish your PhD?* At last I can finally change my answer. Among my former colleagues, a special mention goes to Stefano Izzo, who provided the initial significant support in setting up the technical solution. More than once we stayed late in the office for working and brainstorming sessions. Among my current colleagues, I am grateful to Charuta Pande for her very kind and highly competent support about technical aspects of AOAME. It has always truly been a pleasure to exchange opinions with her, as well as helping me to increase my technical knowledge. My deep appreciation goes also out to the remaining, but not less important, colleagues from the Intelligent Information Systems Research Group: Andreas Martin, Barbara Thönssen, Devid Montecchiari, Hans-Friedrich Witschel, Maja Spahic and Stephan Jüngling. Collaboration with them is awesome.

I would also like to say a heartfelt thank you to my family back in Italy - especially my Mum Grazia, Dad Omar and Grandmother Amalia - for their unconditional love and I apologize for my long periods of absence.

Last but not least, I would like to thank all my friends for their genuine and affectionate friendship throughout the years, with a special mention to George Gontikas and Bugra Ablak.

*To my Grandfather Quinto,
who taught me about courage, resilience and kindness*

ABSTRACT

Domain-Specific Modelling Languages (DSMLs) offer constructs that are tailored to better capture the representational needs of an application domain. They provide customized graphical notations, which facilitate understanding of models by domain experts. As a result, DSMLs allow the construction of domain-specific models that support collaboration, improve work processes and enhance decision-making. Given their special purpose, however, a DSML has to be built each time a new application domain is to be addressed, which is quite demanding and time-consuming. A valid alternative is the creation of DSMLs through domain-specific adaptations of existing modelling languages. This solution has the benefits of starting from a baseline of well-known concepts, which is adapted to fit a specific purpose. Current engineering processes for building or adapting modelling languages, however, lack agility. It follows a sequential engineering lifecycle, where modelling and evaluation activities cannot start before the DSML is deployed for use. Such a sequential approach tends to keep the language engineer separate from the domain expert, who is hindered from gaining experience from the DSML until it is created. The separation of the two roles is a threat to the high quality of the DSML as it requires the joint effort of both experts. On the other hand, the new requirements that arise from the suggestions of the domain expert have to go through the whole engineering lifecycle (i.e. capture and document the requirement, conceptualise, implement and deploy), which is time-consuming. These current drawbacks of present engineering processes have been explored in two case studies, which report the development of a DSML for Patient Transferal Management and a DSML for Business Process as a Service.

In this research an agile meta-modelling approach has been conceived to address the identified drawbacks. Specifically, the approach allows the quick interleaving of language engineering, modelling and evaluation activities. Hence, the close cooperation between the language engineers and the domain experts is fostered from an early stage. A set of operators are proposed to enable on-the-fly domain-specific adaptations of modelling languages, thus avoiding the sequential engineering phases. This agile meta-modelling aims to promote both the high-quality and quick development of DSMLs through domain-specific adaptations. Moreover, to avoid misinterpretation of the meaning of the newly created modelling constructs as well as ensuring machine interpretability of models, the agile meta-modelling has been supplemented by an ontology-aided approach. The latter embeds the specification specifications of modelling languages into an ontology. A set of semantic rules are proposed to support the propagation of language adaptations from the graphical to the machine-interpretable representation. In turn, the approach was developed in the modelling environment AOAME, which allows preserving consistency between the graphical and the machine-interpretable knowledge while domain-specific adaptations are performed. An evaluation strategy is proposed, from which three criteria were derived to evaluate the approach. Firstly, the correct design of the approach is evaluated by the extent to which it satisfies the requirements. Secondly, the operationability of the approach is evaluated by its ability to preserve consistency between the graphical and the machine-interpretable representations. Thirdly, the generality of the approach is evaluated by its ability to be applied in different application domains. The evaluation of operationability and generality are supported by implementing real-world use cases in AOAME. Consequently, the approach contributes to the practice in three different application domains, the Patient Transferal Management, Business Process as a Service and Innovation Processes. The scientific contribution of the approach spans research fields of Domain-Specific Modelling Language, Meta-Modelling, Enterprise Modelling and Ontologies.

PUBLICATIONS

The conference papers and books below were prepared and produced as progress was made in the preparation and completion of this thesis. All the work was conducted by Emanuele Laurenzi under supervision of Prof. Dr. Knut Hinkelmann, Prof. Dr. Alta van der Merwe and Prof. Dr. Ulrich Reimer. The additional co-authors in some publications were work colleagues with either master's degree or PhD in Information Systems or Computer Science as well as Master students.

1. Laurenzi, E., Hinkelmann, K., Goel, M., & Montecchiari, D. (2020). Agile Visualization in Design Thinking. In *New Trends in Business Information Systems and Technology*. Springer Berlin / Heidelberg.
2. Laurenzi, E., Hinkelmann, K., & van der Merwe, A. (2018). An Agile and Ontology-Aided Modelling Environment. In R. Buchmann, D. Karagiannis, & M. Kirikova (Eds.), *The Practice of Enterprise Modelling. PoEM 2018*. (pp. 221–237). Vienna: Springer, Cham. https://doi.org/10.1007/978-3-030-02302-7_14.
3. Laurenzi, E., Hinkelmann, K., Izzo, S., Reimer, U., & van der Merwe, A. (2018). Towards an Agile and Ontology-Aided Modelling Environment for DSML Adaptation. In R. Matulevičius & R. Dijkman (Eds.), *Advanced Information Systems Engineering Workshops. CAiSE 2018. Lecture Notes in Business Information Processing* (pp. 222–234). Springer, Cham. https://doi.org/10.1007/978-3-319-92898-2_19.
4. Kritikos, K., Laurenzi, E., & Hinkelmann, K. (2018). Towards Business-to-IT Alignment in the Cloud. In Z. Mann & V. Stolz (Eds.), *Advances in Service-Oriented and Cloud Computing. ESOC 2017. Communications in Computer and Information Science* (pp. 35–52). Springer, Cham. https://doi.org/10.1007/978-3-319-79090-9_3.
5. Hinkelmann, K., Laurenzi, E., Martin, A., & Thönsen, B. (2018). Ontology-Based Metamodeling. In Dornberger R. (Ed.), *Business Information Systems and Technology 4.0. Studies in Systems, Decision and Control*. (pp. 177–194). Springer, Cham. https://doi.org/10.1007/978-3-319-74322-6_12.
6. Laurenzi, E., Hinkelmann, K., Reimer, U., Van Der Merwe, A., Sibold, P., & Endl, R. (2017). DSML4PTM: A Domain-Specific Modelling Language for Patient Transferal Management. In *ICEIS 2017 - Proceedings of the 19th International Conference on Enterprise Information Systems (Vol. 3, pp. 520–531)*. SciTePress. <https://doi.org/10.5220/0006388505200531>.
7. Reimer, U., & Laurenzi, E. (2014). Creating and maintaining a collaboration platform via domain-specific reference modelling. In *EChallenges e-2014 Conference : 29-30 October 2014, Belfast, Ireland*. (pp. 1–9). IEEE.
8. Laurenzi, E. (2014). A Model-Driven Approach to Create and Maintain an Executable Transferal Management Platform. In *Doctoral Consortium - DCSOFT, (ICSOFT 2014)* (pp. 14–20). Vienna, Austria: SciTePress.

Table of Contents

ACKNOWLEDGEMENT	II
ABSTRACT.....	IV
PUBLICATIONS.....	V
1. INTRODUCTION AND MOTIVATION	1
1.1 RESEARCH PROBLEM.....	3
1.1.1 Need for Domain-Specific Modelling Languages	3
1.1.2 Research Problem 1: Lack of Agility in DSML Engineering	4
1.1.3 Research Problem 2: Inconsistency between the Graphical and Machine- Interpretable Representation of Modelling Languages.....	7
1.1.4 Consolidated Research Problems	7
1.2 RESEARCH OBJECTIVES AND RESEARCH QUESTIONS	10
1.2.1 Main Research Objective and Main Research Question.....	10
1.2.2 Research Objectives and Questions	10
1.3 RESEARCH STRATEGY.....	12
1.4 SCOPE OF THE STUDY	12
1.5 THESIS OVERVIEW.....	12
2. LITERATURE REVIEW.....	15
2.1 ENTERPRISE MODELLING AND MODELS	16
2.2 MODELLING LANGUAGE DEFINITION.....	18
2.3 META-MODELLING	19
2.4 META-MODELLING FOR ENTERPRISE MODELLING LANGUAGES ...	21
2.5 GENERAL-PURPOSE AND DOMAIN-SPECIFIC MODELLING LANGUAGES	23
2.5.1 DSML vs GPML: A Practical Example.....	25
2.5.2 Terminology Definitions	26
2.6 APPLICATION AREAS FOR DSML	28
2.6.1 DSML for Software Development.....	28
2.6.2 DSML for Knowledge Retention	28
2.7 ROLES IN DSML.....	31
2.8 APPROACHES FOR DSML ENGINEERING	33
2.8.1 Advantages of Domain-Specific Adaptation.....	33
2.8.2 Strengths and Weaknesses of Common Techniques for Domain-Specific Adaptations	35
2.8.3 Reflections and Considerations.....	39
2.9 DSML ENGINEERING: MAIN CHALLENGES	40
2.10 DOMAIN-SPECIFIC MODELLING LANGUAGE ENGINEERING LIFECYCLES.....	42
2.10.1 Domain-Specific Language Engineering Lifecycles.....	42
2.10.2 DSML Engineering Lifecycles.....	44
2.10.3 Advances of DSML Engineering Lifecycles	46
2.10.4 Agile Principles for DSML Engineering Lifecycles and Considerations	48

2.11	AGILITY IN ENTERPRISES: DEFINITIONS, PRINCIPLES, TRENDS AND NEW CHALLENGES	52
2.12	MACHINE-INTERPRETABLE KNOWLEDGE AND ENTERPRISE MODELLING.....	55
2.12.1	Ontology Definition	55
2.12.2	Prominent Ontology Languages: RDF(S) and OWL.....	56
2.12.3	Ontologies in Enterprise Modelling	57
2.12.4	Approaches combining Ontologies with Modelling Languages	58
2.13	CONCLUDING SUMMARY	60
3.	METHODOLOGY.....	61
3.1	PHILOSOPHICAL UNDERPINNING – LAYER 1.....	63
3.2	INDUCTIVE RESEARCH APPROACH – LAYER 2	66
3.3	RESEARCH STRATEGY – LAYER 3	67
3.3.1	Design Science Research	67
3.3.2	Design Science Research in this Research Work.....	71
3.3.3	Case Study Research	75
3.3.4	Case Study Research in this Research Work	76
3.4	METHODOLOGICAL CHOICE – LAYERS 4.....	77
3.5	TIME HORIZON – LAYER 5	79
3.6	DATA COLLECTION AND DATA ANALYSIS – LAYER 6	79
3.6.1	Source Triangulation in the Case Study Research	81
3.6.2	Source Triangulation in the Design Science Research.....	82
3.6.3	Literature Review	83
3.6.4	Interviews	83
3.6.5	Data Analysis	85
3.6.6	Ethics of the Research	87
3.7	CONCLUDING SUMMARY	87
4.	PROBLEM RELEVANCE AND CASES	88
4.1	CASE 1: PATIENT TRANSFERAL MANAGEMENT.....	90
4.1.1	Introduction and Motivation.....	90
4.1.2	Methodology	91
4.1.3	Create Phase	92
4.1.4	Design Phase	121
4.1.5	Develop Phase	138
4.1.6	Deploy/Validate Phase	142
4.2	CASE 2: BUSINESS PROCESS AS A SERVICE (BPAAS).....	150
4.2.1	Introduction and Motivation.....	150
4.2.2	Related Work.....	151
4.2.3	Methodology	152
4.2.4	BPaaS Design Environment Architecture	152
4.2.5	Create, Design, Formalize, Develop and Deploy/Validate Phases	153
4.2.6	Engineering Lifecycle Iterations for BPaaS DSML.....	160
4.3	PROBLEMS HINDERING AGILITY IN DSML ENGINEERING AND MAIN CHALLENGES	185
4.4	NEEDS FOR DOMAIN-SPECIFIC ADAPTATIONS: EXPERTS INTERVIEWS	188

4.4.1	Interviewees Selection and Expertise.....	188
4.4.2	Interview Approach.....	189
4.4.3	Interviews Results	190
4.4.4	Three Complexity Levels of Modelling Language Adaptations.....	194
4.4.5	Requirements for Domain-Specific Adaptations of Abstract Syntax and Notation	196
4.5	CONCLUDING SUMMARY	197
5.	AN AGILE AND ONTOLOGY-AIDED META-MODELLING APPROACH....	198
5.1	AN AGILE META-MODELLING APPROACH.....	199
5.1.1	Comparison of an Agile Meta-Modelling Approach with Profile Mechanisms for Domain-Specific Adaptations	200
5.1.2	An Example of the Tight Cooperation in the Integrated Component	201
5.1.3	Towards the Conceptualisation of On-the-Fly Domain Specific Adaptations in the Integrated Component.....	202
5.1.4	Semantic Mapping of Abstract Syntax to Domain Knowledge	203
5.1.5	Requirements for the Domain-Specific Adaptations of Semantics.....	204
5.1.6	Operators for Domain-Specific Adaptations On-the-Fly	205
5.2	TOWARDS A MACHINE-INTERPRETABLE SEMANTICS FOR THE AGILE META-MODELLING	212
5.2.1	Semantic Lifting	212
5.2.2	Problems of Semantic Lifting in the Agile Meta-Modelling	213
5.3	AN ONTOLOGY-AIDED APPROACH FOR THE AGILE META- MODELLING.....	216
5.3.1	The Ontology-based Meta-Modelling	216
5.3.2	Ontology Architecture for an Agile and Ontology-Aided Meta-Modelling Approach	219
5.3.3	Designing Semantic Rules for the Propagation of Domain-Specific Adaptations	229
5.3.4	Syntactic and Semantic Validation of Semantic Rules	232
5.4	CONCLUDING SUMMARY	261
6.	THE AGILE AND ONTOLOGY-AIDED META-MODELLING ENVIRONMENT (AOAME)	262
6.1	THE AOAME ARCHITECTURE.....	263
6.1.1	Technological Solution for the Triplestore	265
6.1.2	Technological Solution for the Graphical User Interface	266
6.1.3	The Model-View-Controller Design Pattern for the AOAME Architecture	267
6.2	THE PALETTE	269
6.2.1	The Three-Step Approach to Populate the Palette	269
6.3	AOAME'S MAIN FEATURES FOR ON-THE-FLY DOMAIN-SPECIFIC ADAPTATIONS.....	275
6.3.1	Feature 1: Extending Modelling Constructs	277
6.3.2	Feature 2: Editing Modelling Constructs	286
6.3.3	Feature 3: Hiding Modelling Constructs	290
6.3.4	Feature 4: Deleting Modelling Constructs	291
6.4	CONCLUDING SUMMARY	292
7.	EVALUATION OF THE AGILE AND ONTOLOGY-AIDED META- MODELLING APPROACH	293

7.1	EVALUATION IN DESIGN SCIENCE RESEARCH	294
7.2	EVALUATION STRATEGY FOR THE ARTEFACT	298
7.3	EVALUATING THE CORRECT DESIGN OF THE ARTEFACT	300
7.4	EVALUATING THE OPERATIONABILITY AND GENERALITY OF THE ARTEFACT	305
7.4.1	Validation of the Functionality that Integrates Modelling Languages.....	308
7.4.2	Validation of Functionalities for Creating New Modelling Constructs, Domain Ontology Concepts and Semantic Mappings	313
7.4.3	Validation of Functionalities for Creating New Bridging Connectors and Datatype Properties	324
7.4.4	Validation of Functionalities for Deleting Modelling Constructs and Properties	334
7.4.5	Validation of Functionalities for Editing Modelling Constructs and Properties	342
7.4.6	Validation of Functionality for Hiding Modelling Constructs.....	350
7.4.7	AOAME for Innovation Processes.....	352
7.5	EVALUATION CONCLUSION AND CONSIDERATIONS	354
8.	CONCLUSION AND FUTURE WORK.....	356
8.1	SUMMARY RESULTS.....	358
8.1.1	Problems Hindering Agility in Domain-Specific Adaptations (Research Question 1).....	358
8.1.2	Needs for Domain-Specific Adaptations (Research Question 2).....	359
8.1.3	Fostering Agility in Domain-Specific Adaptations (Research Question 3).....	359
8.1.4	Automating the Agile and Ontology-Aided Meta-Modelling Approach (Research Question 4).....	361
8.2	CONTRIBUTION.....	362
8.2.1	Artefact and Sub-Artefact Contributions	362
8.2.2	Contribution to Practice	366
8.2.3	Contribution to the Body of Knowledge	367
8.3	METHODOLOGICAL SUITABILITY	368
8.4	RESEARCH LIMITATIONS, EXCLUSIONS AND FUTURE DIRECTIONS	370
8.5	CONCLUDING SUMMARY	371
	BIBLIOGRAPHY	372
	ABBREVIATIONS AND ACRONYMS	391
	LIST OF FIGURES	393
	LIST OF TABLES	400
	APPENDIX A: PATIENT TRANSFERAL MANAGEMENT DOCUMENTATION ..	403
	APPENDIX B: BUSINESS PROCESS AS A SERVICE DOCUMENTATION.....	405
	APPENDIX C: MODELLING EXPERT INTERVIEWS DOCUMENTATION.....	407
	APPENDIX D: VALIDATION SPARQL RULES DOCUMENTATION.....	408
	APPENDIX E: PROTOTYPE DOCUMENTATION	409

APPENDIX F: EVALUATION DOCUMENTATION410

1. INTRODUCTION AND MOTIVATION

Enterprise Engineering is “the body of knowledge principles and practices to design an enterprise” (Giachetti, 2010). An enterprise in the context of Enterprise Engineering is defined as a “complex socio-technical system that comprises interdependent resources of people, information, and technology that must interact with each other and their environment in support of a common mission” (Giachetti, 2010).

The research discipline Enterprise Modelling is the part of Enterprise Engineering that supports the design of enterprises through models (also known as *enterprise models*) and ultimately leverage value for enterprises (Vernadat, 2003; Frank, 2014a; Braun, 2016). Models are means to abstract the complexity of an enterprise in order to focus on relevant aspects, e.g. business processes, business decisions, information systems. The abstraction facilitates interpretation towards better decision making, which ultimately adds value to enterprises. For example, enterprise architecture models support decision makers in business transformation (Zachman, 2008). Decision making is also supported by automation and the information and conclusions gained from models, i.e. analysis, simulation and recommendations. For example, in Business Process Management models are used to improve organisations by reducing costs, execution time and error rates (Dumas et al., 2018).

There exist various enterprise modelling languages that provide sets of pre-defined constructs from which enterprise models can be created. Standard modelling languages such as BPMN (OMG, 2011) for business process modelling, CMMN (OMG, 2016a) for case modelling, DMN (OMG, 2016b) for decision modelling, ArchiMate (The Open Group, 2017) for enterprise architecture modelling, offer modelling constructs that aim for broadly known global consensus. Although such languages bring the benefit of creating uniform and sharable models across enterprises, they are not sufficiently expressive to address every application domain.

As an example, such inadequacy was manifest in the Business Process as a Service (BPaaS) case, conceived in the European project CloudSocket¹ (Woitsch & Utz, 2015). The objective of BPaaS was to bring whole business processes into the cloud. A model-based approach aimed to align business requirements with cloud offerings, which are typically specified in IT language. BPMN 2.0 was adopted to model the business processes (e.g. invoicing business process), but it was not sufficiently expressive to model the business requirements and IT specifications of cloud offerings.

A similar problem arose in the patient transferal management case addressed in the Swiss research project Patient-Radar² (Reimer & Laurenzi, 2014). The objective of the patient transferal management case was to make the transfer of patients from hospital to other sites of post-treatment more efficient. To achieve this objective, a model-based approach aimed to provide all the relevant concepts (events, activities and decisions) of the underlying application domain to all the involved stakeholders. It highlighted the existing modelling language’s lack of specific elements such as hospital-related documents or activities. Besides, available modelling languages were too complex for physicians and transferal managers (i.e. domain experts) to understand. The BPMN 2.0 specification, for instance, spans more than 500 pages and the definition of elements is distributed across different sections and sometimes even with conflicting semantics (Natschläger, 2011).

In both examples, the lack of domain-specific aspects can be overcome by adapting a modelling language. However, the adaptation of a modelling language is not a one-off

¹<https://site.cloudsocket.eu/>

²<https://www.fhsg.ch/en/projects/project/patient-radar-226/>

engineering process. Since domain-specific aspects reflect the underlying enterprise reality, they are subject to continuous changes. Reasons for changes are mainly due to the need to accommodate new requirements that arise from stakeholders' wishes (Karagiannis, 2018), from a better understanding of the domain (Götzinger et al., 2016), or from the targeted application domain such as new business strategy (e.g. due to mergers and acquisitions) leading to a new enterprise architecture, new business processes as well as new regulations and policies. For instance, the recent European law General Data Protection Regulation (GDPR), led companies to include new concepts for the treatment of personal data in their business processes (Robol et al., 2017). The inability to provide such information within a time limit would have resulted in a huge fine for enterprises. That is, enterprises do not only face the challenge of ever-changing requirements but are also under pressure to deliver quickly. According to Horkoff et al. (2018), enterprises are increasingly under pressure due to the current challenging environment, i.e. high competition and cross-organisation cooperation.

In order to keep up with the fast pace, ever-changing and domain-specific requirements should be continuously accommodated in models, in a timely manner while keeping the high quality of models. This is a major challenge in Enterprise Modelling and recalls for new agile and supportive modelling approaches.

1.1 Research Problem

This section presents the research problem, which aims to set the ground for the formulation of the research questions. First the need for creating domain-specific modelling languages (DSMLs) is emphasised. This includes the reasons for integrating domain-specific aspects into modelling languages instead of directly into models. Thus, approaches for creating DSMLs are briefly mentioned and the domain-specific adaptation is motivated. Then, drawbacks of the approach are elaborated leading to the two main research problems tackled in this thesis. The section ends with a consolidated version of the research problem, which follows the writing style and guidelines suggested by Ellis and Levy (2008).

1.1.1 Need for Domain-Specific Modelling Languages

General-Purpose Modelling Languages (GPMLs) allow the modelling of any kind of reality even if the domain is not yet clear. For instance, the Unified Modelling Language (UML) Class Diagram (OMG, 2017) and Entity Relationship Diagram (ERD) can be considered as GPMLs. When adopting GPMLs, basic generic concepts such as “classes”, “relations” and “attributes” are typically used to conceptualise specific enterprise aspects, e.g. an order, a product, an invoice, a customer, a contract. Whereas the domain knowledge is represented in the model the modelling language remains general. Therefore, domain-specific concepts are created through general-purpose concepts. According to Frank (2010), this requires great skills as both expertise in modelling and in domain-knowledge are greatly needed every time a conceptualisation is performed. Moreover, modellers have a quite high degree of freedom in creating models. Along with this is the risk of creating inconsistent and ambiguous models, which results in a threat to model quality (Frank 2010). Petre (2013), France and Rumpe (2007) and France et al. (2006), for example stress the complexity and imprecise interpretation of models created with UML. An additional problem concerns the absence of uniformity among models that are built with a GPML, which makes their comparison unfeasible.

To overcome these problems Pérez and Porres (2013), Lodderstedt et al. (2002) and Felfernig et al. (2000) proposed the supply of modelling concepts with semantics such as constraints, i.e. cardinalities, restriction on attribute types and on values. Along with the same lines, the OMG introduced the Object Constraint Language (OCL) standard (OMG, 2014b) to add constraints on UML class diagram concepts.

This approach, however, has the disadvantage of increasing complexity for modellers as they either have to learn a separate language to express all the constraints (e.g. the OCL in UML), or they are hindered by the modelling tool to make certain modifications to a model when this would violate a constraint (Reimer & Laurenzi, 2014). Constraints might be hidden or hardwired in tools (e.g. in the form of software code), thus are difficult to interpret or not accessible for change (Walter et al., 2014; Atkinson, et al. 2015).

A promising alternative that reduces the degree of freedom of the modeller while not increasing complexity is by integrating requirements directly into a modelling language. This can be done with the widely-adopted technique known as meta-modelling (see Section 2.3). That is, domain-specific concepts and constraints are assimilated in the meta-model. A meta-model is a model of a model, i.e. a specification of the elements of the modelling language (Karagiannis et al., 2016). Integrating domain-specific aspects directly in meta-models create modelling languages that are tailored to a specific domain, and are known as Domain-Specific Modelling Language (DSML) (van Deursen et al., 2000; Mernik et al., 2005; Gray et al., 2008; Frank, 2008; Fowler, 2011; Clark et al., 2015).

Concepts of a DSML are typically represented with graphical notations that are familiar to domain experts. It increases understanding of models among the targeted domain experts, and

thus it promotes their involvement in the creation and adaptation of models (see a concrete example in Sub-section 2.5.1). The inclusion of constraints facilitates the modeller during design time and restricts the variety of models, which fosters the creation of uniform models. In turn, DSMLs promote the creation of quality models (Kelly and Tolvanen, 2008). Given the highlighted benefits, DSMLs are increasingly adopted in both research and industry (Braun et al., 2015; Karagiannis et al., 2016; Neumann et al., 2016).

1.1.2 Research Problem 1: Lack of Agility in DSML Engineering

As already mentioned, DSMLs incorporate domain-specific aspects. Therefore, DSMLs carry a certain degree of domain specificity or domain semantics. It is a challenge to find an appropriate degree of domain specificity (Frank, 2008; Frank 2010; Frank, 2013a; Zečević et al., 2017).

On the one hand, the language must be specific enough to cover the relevant aspects of an application domain. This need can be addressed by developing a dedicated DSML from scratch, where the modelling constructs are completely newly built (Burwitz et al., 2013). On the other hand, there might also be the need to re-use the DSML in other application scenarios with a different degree of domain specificity. In this case, a new dedicated DSML must be re-built for addressing a new purpose. Moreover, instead of re-inventing the wheel, it is convenient that a language benefits from broadly established experiences and lessons learned from an existing modelling language.

The domain-specific adaptation approach allows the combination of all the listed needs (Jablonski et al., 2008; Chiprianov et al., 2012; Atkinson et al., 2013). The approach foresees existing modelling languages, which provide a baseline of concepts with well-known notations and a widely accepted semantic. This baseline can then be further adapted to cover others of a more specific domain. Recent publications show increasing interest on the adoption of the domain-specific adaptation approach, e.g. (Becker, 2014; Braun et al., 2015; Braun et al., 2016; ; Hinkelmann et al., 2016; Karagiannis et al., 2016; Neumann et al., 2016; Zečević et al., 2017; Laurenzi et al., 2017). The domain-specific adaptation refers to the practice of extending or customizing existing modelling languages to tailor them to the needs of an application domain. In this sense, the approach allows increasing the degree of domain specificity of an existing language as domain-specific requirements are incorporated (Jablonski et al., 2008).

There exist different forms of domain-specific adaptations and the most common are known as in-built, model annotation, ad-hoc customisation and multi-level modelling (Atkinson & Kuhne, 2003; Atkinson & Kühne, 2008; Del Fabro & Valduriez, 2009; Robert et al., 2009; Atkinson et al., 2013; Braun & Esswein, 2014; Frank, 2014b; Atkinson et al., 2015; Braun, 2015a; Braun, 2015b; Salehi et al., 2016). Section 2.8.3 provides a comparison of the different approaches. It has been found that the ad-hoc approach has several advantages. In particular, it supports agility by offering a higher degree of freedom in adapting the meta-model, being more intuitive and not restricted to dedicated tools. This results in a wider adoption in both research and industry. In this research, domain-specific adaptations of modelling languages refer to the ad-hoc customisation approach.

The development of DSML through domain-specific adaptations, however, still faces several problems (Braun et al., 2015). Braun et al. (2015), Bork and Fill (2014) and Frank (2013a) stress the still scarce availability of methods, guidelines and best practices for developing a DSML. Heitkötter (2012) and Cho et al. (2012) claim that the design of a meta-model for a DSML remains a crucial task even for language engineers with high expertise. In fact, the development and adaptation of DSMLs, requires both expertise in language engineering and domain knowledge, and few people have both (Mernik et al., 2005; Cho et al., 2012; Chiprianov et al., 2013). The domain knowledge mostly resides in the minds of domain

experts and needs to be extracted by the language engineer. Barišić et al. (2018) and Izquierdo et al. (2013) suggest solutions to foster cooperation between the language engineers and the domain experts or end-users from an early stage of the DSML engineering lifecycle.

On the one hand such solutions improve the quality of the final DSML. On the other hand, they are a quite time-consuming engineering effort. Typically, engineering a DSML is done in several phases (Ceh et al., 2011; Cho et al. 2012). During these phases two prominent components are distinguished: the *language engineering* component and the *modelling* component (Bork et al., 2019). In the language engineering component, the domain analysis is conducted first. Then, the DSML is conceptualised according to the requirements elicited from the domain analysis. Next, the DSML is implemented, and deployed into a modelling tool. A modelling tool implements the modelling component, in which the evaluation and model creation takes place.

Such subsequent chain of phases reverts to the waterfall methodology of software development: a follow-up phase cannot start before the previous one has been completed. In software engineering, the sequential issue was problematic as it resulted in the inability to keep up with the continuous demand for rapidly delivering software. Such inability was the prominent reason for the advent of agile-based methodologies like SCRUM (Schwaber & Beedle, 2002). Similarly, the rigid and sequential phases in DSML engineering lifecycles hinder the quick development (and versioning) of DSMLs.

In response to this lack of agility in DSML engineering, the AMME Lifecycle (also known as OMiLab lifecycle) was proposed in (Karagiannis, 2015; Karagiannis, 2018). The AMME Lifecycle foresees feedback channels along different engineering phases. Channels support agility as are able to interweave between engineering phases. Thus, DSMLs can be developed incrementally.

A closer look to the AMME Lifecycle (Figure 1) reveals that there is still distinction between the language engineering component (engineering cycle on the left-hand side of Figure 1) and the modelling component (evaluation cycle on the right-hand side of Figure 1). Although this approach promotes the correct creation of a DSML, the above-mentioned waterfall-like approach manifests again. A model cannot be designed until a modelling language is deployed (see left-hand side of Figure 1) and, thus the evaluation cannot start (see right-hand side of Figure 1). The separation of the two components can, therefore, be harmful for the quick adaptation of a DSML. In fact, the component for language engineering is dedicated to the language engineers while the component for the evaluation is dedicated to the end-users (i.e. modellers and domain experts). As above stated and reported by Barišić et al., (2018) and Izquierdo et al., (2013), such separation is not ideal in the creation of DSMLs as it tends to hinder the collaboration between the language engineer and the end-users, which is a threat for the quality of the resulting DSML.

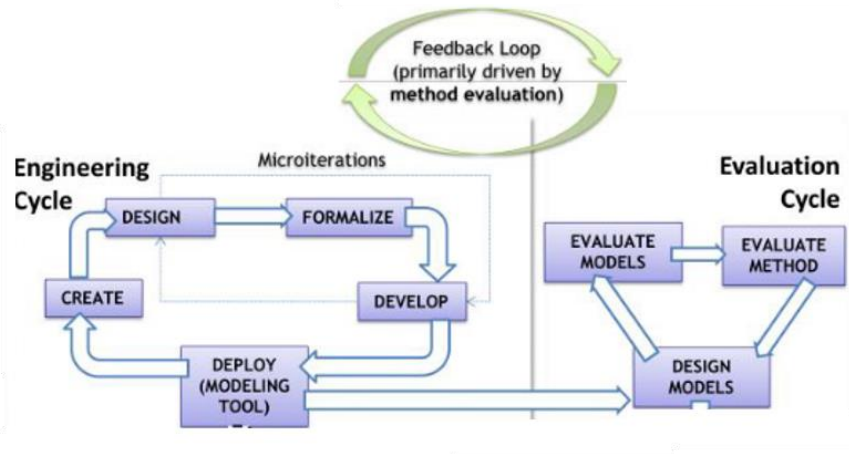


Figure 1. The AMME Lifecycle (extracted from Bork et al., 2019)

These problems show the need to further address the agile principles to quickly engineer high quality DSMLs. According to the pioneers of agile principles - Agile Manifesto (Beedle et al., 2001) for software engineering and Burlton et al., (2017) for business development - an agile approach should

- avoid sequential phases while being incrementally engineered,
- foster collaboration with end-users from an early stage of each engineering lifecycle.

Specifications of modelling standards like UML, BPMN, and ArchiMate foresee approaches for an “on-demand” extension, such as those which are employed in modelling tools. These approaches seem to embrace the two above-mentioned agile principles, as one can add new modelling constructs, attribute types and values on-the-fly into a sort of joint component *engineering-modelling*. As an example, the tool Visual Paradigm³ implements profiling mechanisms like *stereotype* and *tagged values* to adapt both modelling concepts and data structure of ArchiMate. Hence, such an adaptation has no sequential engineering phases and cooperation with end-users can be fostered by the joint component. In contrast, however, it does not allow ad-hoc customisations of meta-models. In fact, this adaptation approach is known as *built-in* and due to the several limitations and drawbacks (e.g. inflexibility and abstraction conflicts) is highly criticised by many authors (Robert et al. 2009, Atkinson et al., 2013; Braun & Esswein, 2014; Braun, 2015a; Braun, 2015b; Salehi et al., 2016).

The two identified agile characteristics from the built-in approaches are, however, valid concepts and could be transferred into an ad-hoc customisation approach. That is, domain-specific adaptations could be performed on-the-fly in a joint component *engineering-modelling*. As in the built-in approaches, the sequential phases would be avoided and cooperation between different experts would be supported, resulting in the quick creation of high quality DSMLs.

It is the first objective of this research to conceive an agile approach that allows on-the-fly domain-specific adaptations in a joint component for both engineering and modelling.

Achieving this objective, however, opens an additional research problem, and one which is presented in the following sub-section.

³ <https://www.visual-paradigm.com/>

1.1.3 Research Problem 2: Inconsistency between the Graphical and Machine-Interpretable Representation of Modelling Languages

A new modelling construct comes with a new meaning. The meaning is typically implicit in a graphical representation or notation of a modelling construct. Sometimes, the meaning of a modelling construct is made explicit in a separate description using natural language, which is considered an informal specification. The explication of the meaning aims to increase a shared understanding of the new modelling construct, e.g. its context, its purpose and perhaps how to use it when creating models.

An informal specification of the meaning, however, is open to subjective interpretations. This is in contrast to the aim of DSMLs to foster shared understanding among the stakeholders (Selic, 2007). The specification of the meaning in a meta-model increases its degree of formality. According to Harel and Rumpe, (2004) a formal specification of semantics makes its interpretation unambiguous. Bork and Fill (2014a) argue that grounding modelling constructs with formal semantics enables intersubjective understanding. Ontology languages such as RDF(S) or OWL (Allemang & Hendler, 2011), allow specifying semantics in a machine-interpretable representation, which brings the benefit of powerful automation (Hinkelmann et al., 2016).

A common approach that provides a machine-interpretable representation is called semantic lifting (Kappel et al., 2006; Azzini et al., 2013), also known as semantic annotation (Fill et al., 2013; Bork & Fill, 2014; Liao et al., 2015). On the one hand, there is the graphical representation of the modelling construct, i.e. the graphical notation. On the other hand, there is an ontology, which is the machine-interpretable representation of the modelling construct. The two representations are associated with each other in a relationship that provides graphical notations with formal semantics (Hrgovic et al., 2013).

Although associated with each other, the graphical and the machine-interpretable representations are strictly separate. This is problematic when performing domain-specific adaptations as inconsistency issues arise between the two representations if these are not kept aligned (Hinkelmann et al., 2016). When a change occurs in one of the two representations, the other has to be adapted accordingly. For instance, when inserting a new modelling concept the correspondent ontology concept has to be entered. This alignment is mainly done manually or semi-automatically, which is not only error-prone and time-consuming, but also requires ontology expertise. Although there exist mechanisms for the automatic generation of ontologies from models, e.g. (Emmenegger et al., 2017; Karagiannis & Buchmann, 2018), they still require manual actions to perform transformation from the graphical to the machine-interpretable representation, which again can be time-consuming and cause inconsistencies in cases when transformations are not made after some change.

It is the second objective of this research to ensure seamless consistency between the graphical and the machine-interpretable representation of a modelling language while domain-specific adaptations are performed.

1.1.4 Consolidated Research Problems

The research is motivated by the inability of current approaches to keep up with the continuous demand for quickly accommodating domain-specific modelling requirements into models (*Step 1* in Figure 2).

Domain-specific adaptations allow integrating these requirements directly in the modelling language. The approach promotes the creation of models with higher quality compared to the domain-specific models created with GPMLs (Frank, 2013a). Additionally, the adaptation of existing modelling languages is preferable over the creation of DSMLs from “scratch”, because

it comes with a baseline of known graphical notations and semantics. Therefore, it promotes shared understanding of a DSML. The model-based approach adopted to perform the domain-specific adaptations is meta-modelling, also known as ad-hoc customisation of meta-models. Meta-modelling requires high expertise in both the application domain and in language engineering (Frank, 2013a; Bork & Fill, 2014; Braun et al., 2015), but a few people have both (Cho et al., 2012; Heitkötter, 2012). A lack of one of expertise in one of the two fields is a threat to the quality of the resulting DSML (Frank, 2013a). To ensure the inclusion of both expertise Barišić et al. (2018) and Izquierdo et al. (2013) suggest starting cooperation between language engineers and domain experts from an early engineering phase of the DSML. This approach, however, supports agility only partially as a DSML is built through different sequential phases, which are typically conceptualise, implement and evaluate. This waterfall-like approach shows a lack of agility (*Step 2* in Figure 2) in DSML engineering.

The AMME lifecycle (Karagiannis, 2015) takes agility a step forward by foreseeing feedback channels along with the different phases of the lifecycle, and several DSMLs were developed supported by the latter (Karagiannis et al., 2016). However, it still distinguishes between the language engineering and the modelling component, which address two different expertise. The two-component approach reverts to two main issues:

- The cooperation between the language engineers and domain experts is hindered because of the use of different components (engineering and modelling), which keep separate the engineering from the modelling and evaluation activities;
- The accommodation of new requirements in the DSML is time-consuming. Each new requirement still needs to go through the different engineering phases, sequentially.

Ideally, an agile meta-modelling approach should (*step 3* in Figure 2):

- Promote cooperation between the language engineer and domain expert from an early engineering phase.
 - o This challenge can be addressed by allowing the interleave of language engineering, modelling and evaluation activities in one single joint component;
- Incrementally develop DSMLs while avoiding sequential engineering phases.
 - o This challenge can be addressed by allowing *domain-specific adaptations* in the joint component *on-the-fly*.

Approaches that incorporate these two agile principles exist such as profiling mechanisms. Such mechanisms are commonly implemented in modelling tools. However, among other drawbacks, they suffer from potential misinterpretation of newly defined concepts (Selic 2007; Battistutti & Bork 2017) and from limited automation (*Step 4* in Figure 2).

Making the meaning of the concepts explicit in a machine-interpretable form would tackle these drawbacks (Emmenegger et al., 2013; Natschläger 2011; Rospocher et al., 2014; Walter et al., 2014; Hinkelmann et al., 2016) (*Step 5* in Figure 2).

Semantic lifting and semantic annotation (Kappel et al., 2006; Azzini et al. 2013; Fill et al., 2013; Hrgovic et al., 2013; Bork & Fill, 2014; Liao et al., 2015) define the meaning of a modelling language using an ontology. However, because of the separation of modelling language and ontology, the correct machine-interpretable semantics is not ensured when performing domain-specific adaptations. Similarly, if a change occurs in the ontology, the graphical modelling language has to be adapted, accordingly. That is, if for each change the graphical or machine-interpretable representation is not adjusted, inconsistencies can be created (Hinkelmann, Gerber, et al., 2016) (*Step 6* in Figure 2).

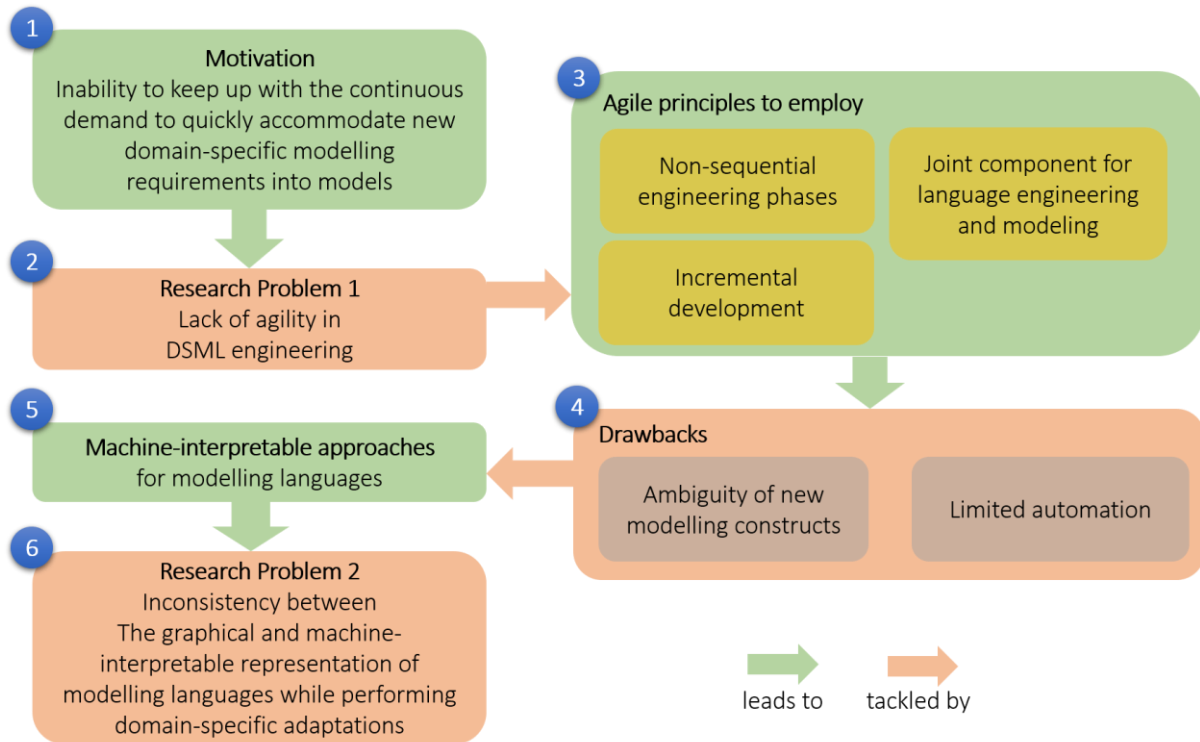


Figure 2. Map of the Consolidated Research Problem

1.2 Research Objectives and Research Questions

In keeping with Creswell (2002) and Ellis and Levy (2008), research objectives point to what the research study intends to do in order to address the problem. Then, research questions operationalize research objectives. Namely, research questions serve to restrict the research objectives into specific questions that should be addressed in the study. The answer to a research question enables meeting the corresponding research objective, thus contributing towards solving the research problem (Leedy & Ormrod, 2005).

Section 1.2.1 presents the main research objective followed by the main research question of this thesis. As suggested in (Cronje, 2011), a constructive approach is used to achieve the answer to the main research question. This consists of the formulation of research questions, which contribute to answering the main research question.

The main research question is broken down into more detailed research objectives and research questions, which contribute to solving the two main research problems identified in the previous section.

1.2.1 Main Research Objective and Main Research Question

The *first research problem* is about the lack of agility in the engineering of DSMLs. As stressed in the research problem, engineering DSMLs through domain-specific adaptation can contribute to the solution of this problem. For this, a machine-interpretable approach addresses the drawbacks facing the new modelling constructs, i.e. ambiguous interpretation and limited automation. Considering a machine-interpretable approach, led to identify the *second research problem* to solve in this work: the occurrence of inconsistencies between the graphical and the machine-interpretable representation of modelling languages during domain-specific adaptations.

In order to contribute towards a solution of the two identified problems, the main research objective (MRO) of this work is to conceive an approach that fosters agility for domain-specific adaptations of modelling languages while preserving the seamless consistency between their graphical and the machine-interpretable representation. The main research objective is operationalized in the main research question as follows:

Main Research Question (MRQ): *How can agility for domain-specific adaptations of modelling languages be fostered while preserving seamless consistency between their graphical and machine-interpretable representations?*

1.2.2 Research Objectives and Questions

The main research objective (MRO) is broken down into four research objectives, which are operationalized into research questions. The answers of all the research questions contribute to answer the main research question (MRQ).

The first research objective (RO1) is to identify the problems that hinder agility in domain-specific adaptations of modelling languages. Thus, the following research question is posed:

1. *Research Question 1 (RQ1):* *What are the problems that hinder agility in domain-specific adaptations of modelling languages?*

To answer this research question first there is the need to increase understanding of how domain-specific adaptations are performed over modelling languages for the creation of

DSMLs. The increased understanding provides the basis for the answer to the first research question (*RQ1*). In turn, challenges to be addressed by the new artefact are derived.

The follow-up research objective (*RO2*) aims to increase understanding of the needs for domain-specific adaptations from practice. This objective is operationalized by research question 2:

2. *Research Question 2 (RQ2): What are the needs for domain-specific adaptations of modelling languages?*

The answer to *RQ2* is used to create a list of requirements for domain-specific adaptations.

Both the above-mentioned challenges and the requirements are considered to meet the next research objective (*RO3*). *RO3* aims to propose an approach that fosters agility when domain-specific adaptations are performed. In turn, the objective is operationalized by research question 3:

3. *Research Question 3 (RQ3): How can agility be fostered when performing domain-specific adaptations of modelling languages?*

The answer to *RQ3* provides a new agile approach that tackles both of the above-mentioned research problems: (1) the lack of agility in DSML engineering, and (2) inconsistency between the graphical and the machine-interpretable representations of modelling languages.

The next research objective (*RO4*) then aims to develop a technical prototype that automates the conceived approach. The prototype should preserve consistency between the graphical and the machine-interpretable representations while domain-specific adaptations are performed on modelling languages. The objective is operationalized through research question 4:

4. *Research Question 4 (RQ4): How can the agile approach for domain-specific adaptations that preserves seamless consistency between the graphical and the machine interpretable representation be automated?*

1.3 Research Strategy

This research work follows the design science research (DSR) methodology proposed by Vaishnavi and Kuechler (2004) (see Chapter 3). The methodology consists of five phases: *Awareness of Problem, Suggestion, Development, Evaluation* and *Conclusion*. To ensure both theoretical foundation and relevance in the practice of the research, the rigor and relevance cycles proposed in (Hevner, 2007) are considered.

Research questions 1 (*RQ1*) is answered in the *awareness of problem* phase. For this, the case study research is adopted, which contributes to a deeper understanding of the problems hindering agility of DSML engineering. Research question 2 (*RQ2*) is also answered in the *awareness of problem* phase. For this, the need for domain-specific adaptations of modelling language is gained from modelling experts' interviews. In the *suggestion* and *development* phases, the research questions 3 (*RQ3*) and 4 (*RQ4*) are addressed, respectively. The main artefact of this research work is conceived in the *suggestion* phase. Its prototypical implementation is then described in the *development* phase. In the *evaluation* phase the artefact is evaluated, which is underpinned by an evaluation strategy. In the conclusion the research results are communicated, and future research is proposed. Section 3.3 provides the theoretical background of the research strategy as well as an extensive description of how it is applied in this research work.

1.4 Scope of the Study

This study deals with the domain-specific adaptation approach for Domain-Specific Modelling Languages (DSMLs) engineering. The approach refers to the meta-model customisation technique, also known as meta-modelling. The study extends to machine-interpretable knowledge approaches to ground DSMLs with ontologies. For this, ontologies and semantic rules are considered. The application domains focus on Enterprise Modelling. For evaluation purposes the scope has been extended to innovation processes.

The elaboration of constraints to further specify semantics of modelling language is left to future work. A more detailed list of limitations and future works is reported in Section 8.4.

1.5 Thesis Overview

Figure 3 graphically depicts an overview of the thesis. The thesis is divided into eight main chapters:

Chapter 1 introduces the motivation of this research work. Next, the research problem description is formulated. The latter is introduced in two versions: (1) the research problem is split into two sub-problems extensively articulated with theoretical underpinnings; (2) The second one is a consolidated version of the research problem. Next, the research objectives and research questions are described, respectively.

Chapter 2 presents the literature review for this research work. It contains the relevant aspects underpinning the formulation of the research problem, research objectives and research questions. It also includes theoretical background needed to support attaining answers for the research questions.

Chapter 3 deals with the research methodology and design. It covers a description of the methodology components selected to conduct this research study, and how they are applicable to it. In this thesis, the research questions and the main research question are answered by adopting the Design Science Research methodology. The methodology is supplied with a case study research to increase understanding of the problem.

Chapter 4 introduces the practical relevance of the research problem. It contains two sections with two different cases “Patient Transferal Management” and “Business Process as a Service”, respectively. The two cases are designed following a case study strategy and are used to answer the first research question (RQ1). Material produced in the creation of the two cases and source documents are reported in two respective Appendixes A and B. The third section of the chapter contains the modelling expert interview results. The interviews are performed to answer to the second research question (RQ2). Sources of the interviews are reported in Appendix C. The two cases are analysed to identify the problems hindering agility of the domain-specific adaptations. The problems are then combined with findings from the literature so to derive the two main challenges to be addressed by the new artefact. The interviews identify the needs for domain-specific adaptations. The needs are then combined with findings from the literature to convey a list of requirements used for the tentative design of the artefact.

Chapter 5 describes the tentative design of an agile and ontology-aided meta-modelling approach. In this chapter the third research question (RQ3) is answered. The challenges and the list of requirements listed in the previous chapter are addressed here. In particular, an agile meta-modelling approach is first conceived, in which the two components for language engineering and modelling are integrated to enable on-the-fly domain-specific adaptations of modelling languages. For the adaptations, a set of operators was proposed. Then, Next, the agile meta-modelling is incorporated into an ontology-aided approach to ensure consistency between the graphical and the machine-interpretable representation while performing domain-specific adaptations. Hence, the specification of a modelling language into an ontology language is provided as well as semantic rules that aid the propagation of the adaptations from the modelling language to the ontology language. The semantic statements are transformed into SPARQL rules.

Chapter 6 describes the technical implementation of the agile and ontology-aided meta-modelling approach. This chapter addresses the fourth research question (RQ4). That is, the approach is instantiated into a working modelling environment named “AOAME”. The source code, ontologies and graphical notations of the prototype are reported in Appendix E.

Chapter 7 deals with the evaluation of the agile and ontology-aided meta-modelling approach. For this, an evaluation strategy is designed, from which evaluation criteria are identified. Next, the approach is evaluated with respect to the criteria and through the use of the AOAME prototype. For the evaluation, real-world use cases are considered. Additional material produced during the evaluation phase as well as a getting started guide is reported in Appendix F.

Finally, *Chapter 8* introduces the conclusion and future work. First, a summary of the research results is provided for each research question. Then, the contribution of the research is reported in terms of practice and body of knowledge. One section shows the adherence of the research work with respect to the chosen methodology Design Science Research. Next, limitations of this research work are discussed as well as early results of the future research directions are reported.

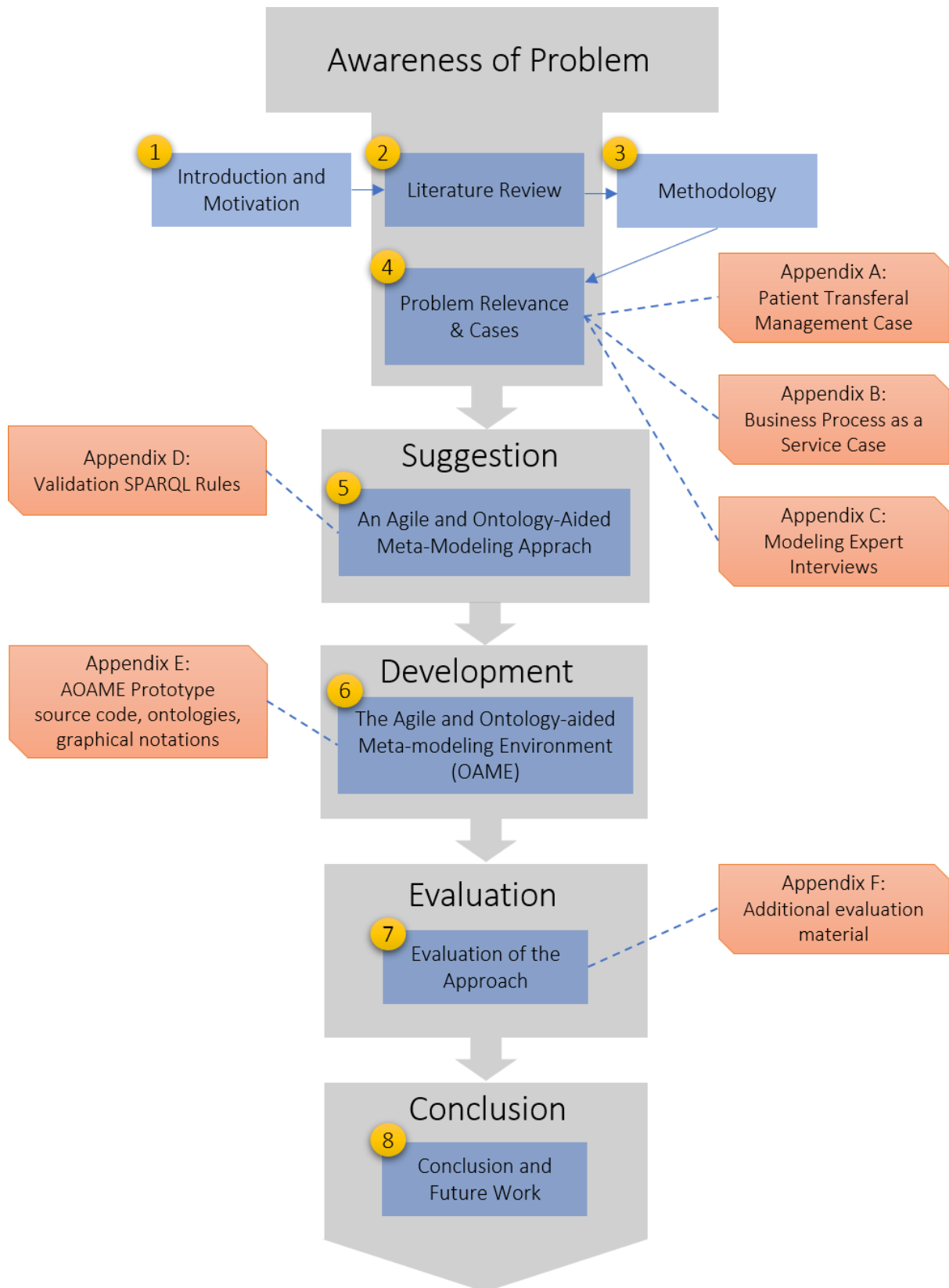
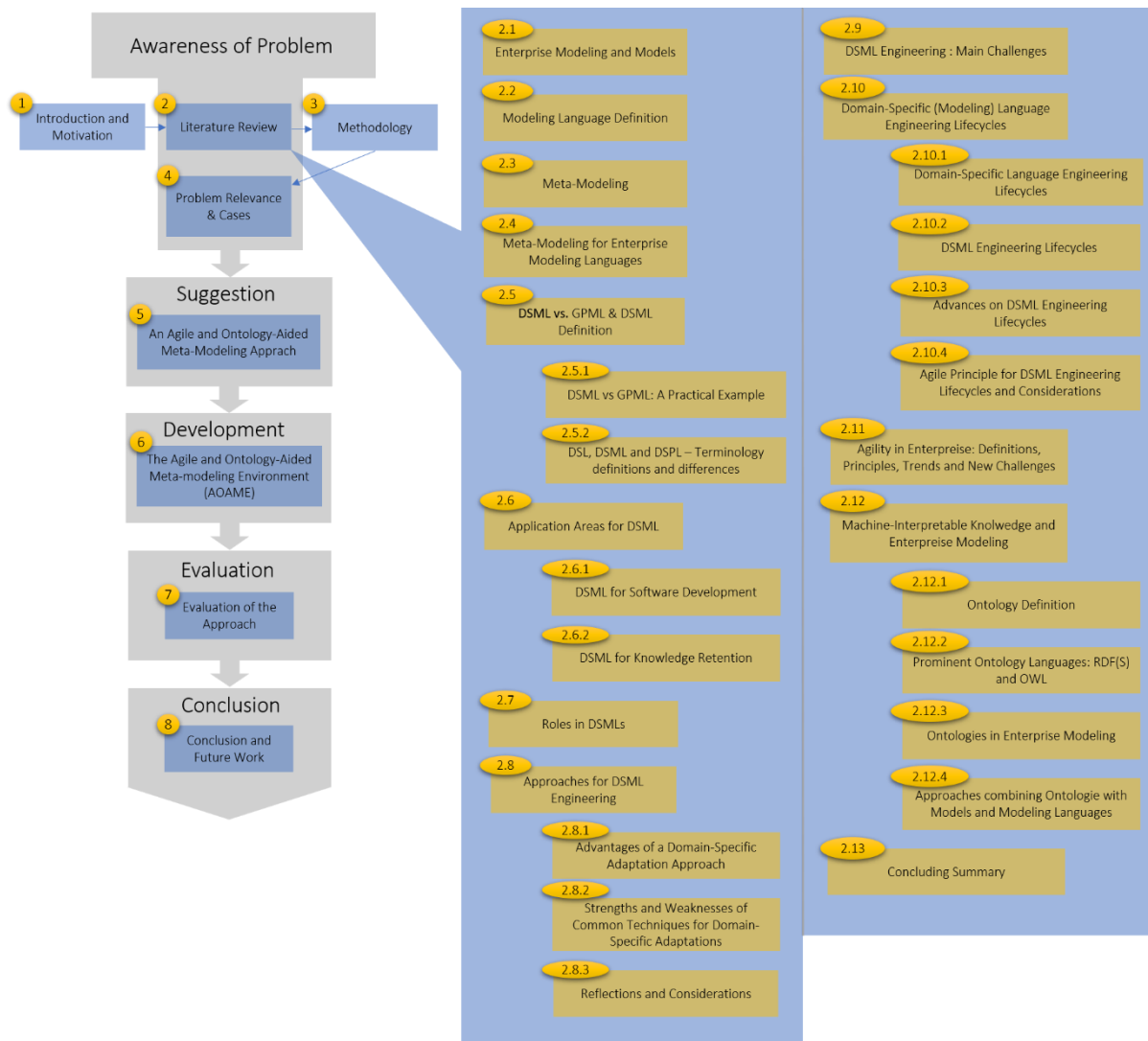


Figure 3. Thesis chapter layout

2. LITERATURE REVIEW



For research projects of an explorative nature, Mouton (2001) suggests to organise the literature in to distinct topics that are relevant to the research. Research always originates with a problem, and the literature review serves as a foundation upon which the research is built (Ellis & Levy 2008). In line with this suggestion, the literature reviewed in this chapter focuses on the relevant topics identified in the problem descriptions.

2.1 Enterprise Modelling and Models

Enterprise Modelling (EM) is a pivotal field in Information Systems (IS) research (Frank, 2014a). An enterprise can be defined as a highly complex heterogeneous socio-technical information system, whose parts are interrelated within a nexus of interdependencies on different abstraction levels (Vernadat, 2003; Braun, 2016). Similarly, Giachetti (2010) defines an enterprise as a “complex socio-technical system that comprises interdependent resources of people, information, and technology that must interact with each others and their environment in support of a common mission”.

EM strives to cope with such a complexity through the creation of models. Benyon et al. (1999) defines a model as “a representation of something, constructed and used for a particular purpose”. The “representation of something” can be further defined as a commonly agreed abstraction, which describes and represents the relevant aspects of a “system under study” (SUS, also known as the “Universe of Discourse”, “subject” or “domain”) (Rodrigues & Silva, 2015; Hinkelmann et al., 2017; Efendioglu et al., 2017).

Many different kinds of models exist, such as graphical models, conceptual models, mathematical models, logical models or formal-based models like enterprise ontologies (Dietz, 2006; Hinkelmann et al., 2013).

According to Vernadat (2003), models are used to capture relevant-enterprise related aspects, which include not just structure and organisation (i.e. static phenomena of Information Systems), but also behaviour, i.e. dynamic phenomena of Information Systems. More specifically, these aspects can relate to activities, processes, information, resources, people, goals, and constraints of a business (Fox & Gruninger, 1998). These are not limited by the boundaries of a particular organisation, for example, an enterprise model may also represent inter-organisational aspects (Frank, 2014b).

Enterprise models such as business processes, organisational models, and enterprise architecture models, can be seen as means through which enterprise knowledge is externalised. The ultimate goal of models is of creating value to an enterprise (Sandkuhl et al., 2018). For this, models support producing a common understanding, inter-subjective communication, documentation, analysis and operations, i.e. answering queries, simulating behaviour, performing reasoning, verification and validation and software generation of software code (Fox & Gruninger 1998; Vernadat, 2003; Fill & Karagiannis, 2013; Bork & Fill, 2014; Hinkelmann, Gerber, et al., 2016; Braun, 2016). Models are also used to support business transformation (Zachman, 2008). This is primary focus of enterprise architecture models, which describe the structural dependencies between different perspectives of an enterprise: organisational, product, business process, data, application and technical infrastructure perspectives (Zachman, 2008).

Bridgeland and Zahavi (2009) describe eight ways of generating business value through models:

- *Communication between people* - models are used to convey complex business information among stakeholders.
- *Training and learning* - models are built on expert’s knowledge and effectively communicate how-to knowledge, i.e. task-by-task on how a job should be performed.
- *Persuasion and selling* - models as a communication means to persuade on taking actions.
- *Analysis of a business situation* - models are analysed to provide insights (e.g. into customer problems), which is of competitive advantage.

- *Compliance management* - models to manage compliance with respect to law, government regulations, and other guidance.
- *Development of software requirements* - models can contain specification of what a software application should do and it does it more effectively than long documents.
- *Direction in software engines* - a model can be used by software to make decisions.
- *Knowledge management and reuse* - models are used to systematically capture enterprise knowledge so that it can be re-used by other people to when needed, e.g. to solve company tasks.

In the Model-Driven Engineering (MDE) (Schmidt, 2006; Rodrigues & Silva, 2015) paradigm, models are transformed into executable specifications for the selected target platform. In this case, models are used to abstract from the complexity of systems by focusing on the problem space rather than the solution space. That is, the abstraction expresses the design in terms of concepts in application domains (healthcare, supply chain management, manufacturing) rather than through the underlying computing environment, e.g. CPU, memory and network device (Schmidt, 2006). The abstraction promises to produce high quality software rapidly, by automatically generating code from models or configuring the behaviour of existing systems (Stahl & Völter, 2006). In support of the MDE paradigm, the Object Management Group (OMG) launched the Model-Driven Architecture (MDA) (Miller Joaquin Mukerji, 2003) paradigm that follows the model-driven approach for the development of software systems. Further implementations of the MDE paradigm are Model-Driven Software Development (MDSD) (Völter et al., 2013) and Model-Driven Development (MDD) (Stahl & Völter, 2006).

The purpose of the model in question determines the requirement for the modelling languages. Modelling language plays a key role as it is used to create models (Selic, 2011). The next section elaborates on the definition of modelling languages.

2.2 Modelling Language Definition

In their framework, Karagiannis and Kühn (2002) define a modelling language with three specifications: notation, syntax and semantics (see Figure 4). There exist two kinds of syntax, one refers to the concrete syntax (or notation) of a language whereas one refers to the abstract syntax. The component labelled as “Syntax” depicted in the Figure 4 refers to the abstract syntax.

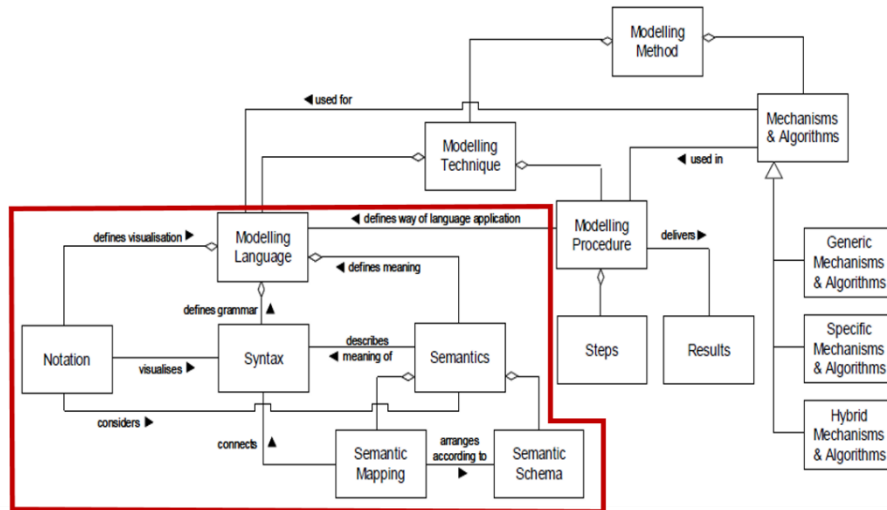


Figure 4. Components of modelling methods. Three modelling language specifications within the red box (Karagiannis & Kühn, 2002)

Abstract syntax refers to the class hierarchy of modelling constructs (or modelling concept) together with their relations and attributes, through which the language terminology is defined. Modelling constructs are syntactic elements typically expressed through a graphical or textual notation. The *notation* is the *concrete syntax* of a modelling language, through which it is possible to create a model. Graphical notations should be cognitively adequate to ensure users’ understanding of models (Hinkelmann et al., 2018).

There is a distinction between a relation in the abstract syntax and the modelling relation expressed through a graphical notation. A graphical notation is also known as visual connector (Karagiannis & Buchmann, 2018) and are used in conjunction with modelling elements to design a model. Thus, both modelling elements and modelling relations can be regarded as modelling constructs.

The *semantics* allows determining the truth value of elements in the model with respect to the underlying reality being conceptualised (Parreiras, 2012). In other words, the semantics defines the meaning of syntactic elements. Semantics can be subdivided into structural and behavioural semantics (Harel & Rumpe, 2000). The structural semantics are specified in the form of structural relations, i.e. class-instance relationships or class-class relationships such as specialisation, generalization, aggregation, composition. Behavioural semantics, on the other hand, specify the behaviour such as the flow of a business process or pre- and post- conditions of a process. Petri nets have a formally specified behavioural semantics.

In this work the focus is on the structural semantics as it regards models as a means to represent knowledge and does not focus on models that represent behaviour. Therefore, in this research work the general term semantics refers to the structural semantics.

2.3 Meta-modelling

Meta-modelling is a model-based technique (or knowledge representation paradigm) commonly used to specify modelling languages (Karagiannis et al., 2016). Meta-modelling distinguishes different levels of abstraction (Völter et al., 2013). Although the number of abstraction levels can be undefined (n^{th} levels), in practice it is common to have three levels. Starting from the top, the language at *level three* is self-represented, meaning there is no need for a higher level of abstraction for its definition. As an example, this can be compared with the Extended Backus-Naur Form (EBNF) notation, which can be realised with a few EBNF statements (Bézivin, 2005). Level three contains the model having concepts and relations (i.e. meta²model) that specify a meta-modelling language. The latter is then used to create models at *level two*, i.e. the meta-model. For instance, in the meta-modelling framework MOF (Meta-Object Facility) (OMG, 2016c), UML elements and relations residing at level three are used to create a meta-model to specify modelling languages such as BPMN, CMMN, DMN. A meta-model is a model of a model. A meta-model contains both concept taxonomy and descriptive properties in the form of a model, which correspond to a modelling language specification, i.e. abstract syntax (including constraints) (Favre, 2005). Thus, the terminology of a modelling language is defined by the meta-model. In graphical modelling languages, elements from the abstract syntax are associated with graphical notation to create graphical models at *level one*. The graphical notations are not just shaped boxes that rely on human-interpretation. Instead, each notation is instantiated from a higher abstraction concept with explicitly defined semantics, which are based on both a concept taxonomy and descriptive properties. Finally, graphical models represent a subject under consideration of an underlying reality, which resides at *level zero*.

Figure 5 presents MOF’s definition for each layer and an example for a model, meta-model and meta-meta-model (OMG, 2016c). As a practical example, the meta-model of UML resides in level two, and the UML class diagram model is in level one (level zero is omitted as it is often referred to the run-time data).

Layer	Description	Example
Meta-metamodel	The infrastructure for a metamodeling architecture. Defines the language for specifying metamodels.	<i>MetaClass, MetaAttribute, MetaOperation</i>
Metamodel	An instance of a meta-metamodel. Defines the language for specifying a model.	<i>Class, Attribute, Operation, Component</i>
Model	An instance of a metamodel. Defines a language to describe an information domain.	<i>StockShare, askPrice, sellLimitOrder, StockQuoteServer</i>
User objects (user data)	An instance of a model. Defines a specific information domain.	<i>{Acme_SW_Share_98789}, 654.56, sell_limit_order, {Stock_Quote_Svr_32123}</i>

Figure 5. Definition of levels by MOF (OMG 2016c)

Figure 6 is an extension of MOF. It distinguishes between meta-model and modelling language (Stahringer, 1996). Namely, a model at level one is created using a modelling language that is described by a meta-model and, in turn, a meta-model is created using a meta-modelling language that is described by a meta²model. Respectively, this implies that a model *conforms to* (and is an *instantiation of*) a meta-model and a meta-model *conforms to* (and is an *instantiation of*) a meta²model (Höffner, 2007).

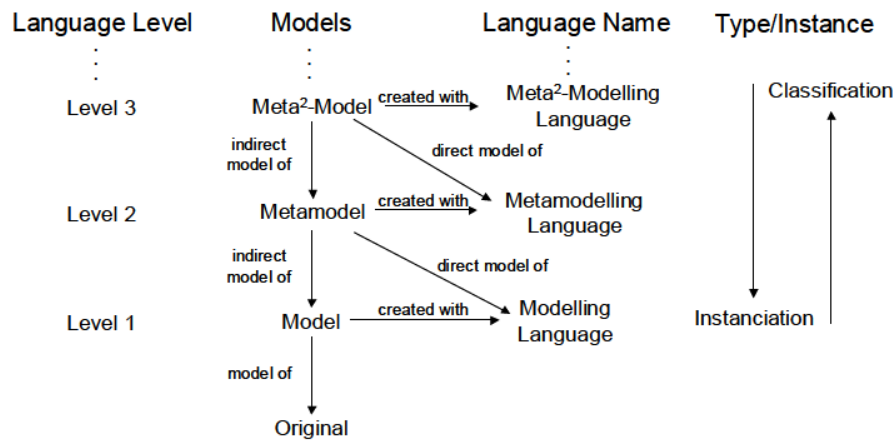


Figure 6. Meta-modelling hierarchy (Strahringer, 1996)

The meta-modelling hierarchy is widely used and in some cases adapted in both academia (Karagiannis & Kühn, 2002; Atkinson & Kuhne 2003) and industry (OMG, 2016c).

2.4 Meta-Modelling for Enterprise Modelling Languages

Enterprise modelling languages (Braun, 2015b) like Archi-Mate (Open Group, 2017), Business Process Model and Notation (OMG, 2011), Case Management Model and Notation (OMG, 2016a), Decision Model and Notation (OMG, 2016b) are typically modelled as UML class diagrams in level two. Figure 7 shows that most of the industry modelling standards from well-known standardisation organisations have UML-class diagram as a graphical mean to represent the abstract syntax.

		Abstract Syntax		
		Graphical	Textual (formal)	Textual informal)
OASIS	Universal Business Language (UBL)	UML-Class Diagram	BNF	Natural Language
	The eBusiness eXtensible Markup Language (eXML) Business Process Specification Schema (BPSS)		XSD	Natural Language
	Web Services Business Process Execution Language (WS-BPEL)	UML-Class Diagram	XSD	Natural Language
OMG	Business Process Model and Notation (BPMN)	UML-Class Diagram	XSD	Natural Language
	Case Management Model and Notation (CMMN)	UML-Class Diagram	XSD	Natural Language
	Decision Model and Notation (DMN)	UML-Class Diagram	XSD; BNF	Natural Language
Open Group	ArchiMate	UML-Class Diagram		Natural Language

Figure 7. Representation of standard modelling languages, adapted from (Efendioglu et al., 2017)

Figure 8 shows the three layers through which a modelling language is defined. The meta-meta-layer presents general concepts and relations of a UML class diagram. These are further specified in the meta-model to represent, in this case, the abstract syntax of BPMN. Concepts in the meta-model are then instantiated in the modelling layer to design a BPMN process model.

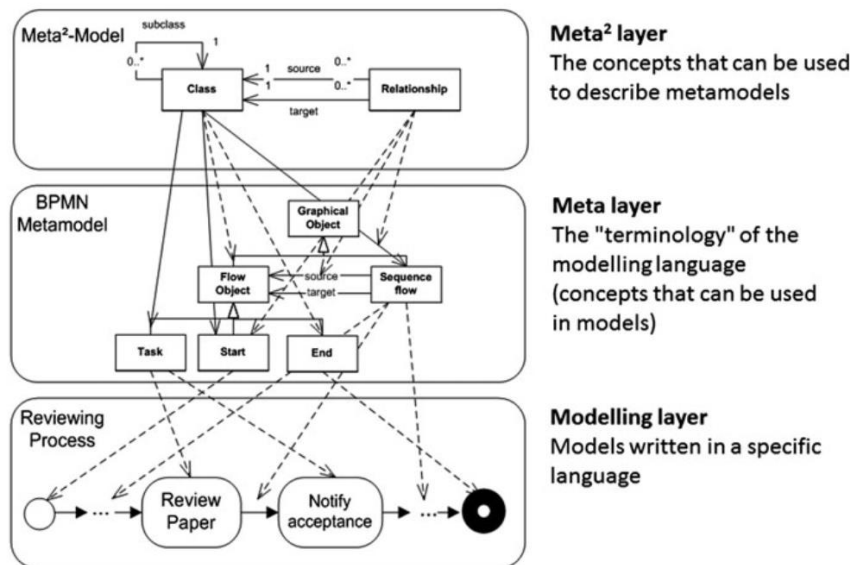


Figure 8. Abstraction layers of BPMN in meta-modelling (Karagiannis *et al.*, 2016)

Adopting the meta-modelling technique provides the benefits of a model-driven approach, namely:

- a) models can abstract away from unnecessary aspects while focusing on the relevant ones of an underlying domain;
- b) the problem domain is better understood while conceptualising it in the meta-model;
- c) a vocabulary for modelling constructs in the considered domain is defined.

While (a) and (b) help reduce the design effort of a modelling language, (c) creates the semantics of a modelling language, which results in tangible artefacts that may be exchanged, inspected, and discussed. The latter enhances the comprehensibility of a modelling language.

When a meta-model contains concepts that capture a particular domain, it can be categorised as a domain-specific modelling language (DSML). For example, the meta layer in Figure 8 contains concepts for process modelling, i.e. task, sequence flow, start and end event. Conversely, when there are no pre-defined concepts and relations in the meta-model layer, any kind of reality can be modelled. In turn, the modelling language can be categorised as general-purpose modelling language (GPML).

2.5 General-Purpose and Domain-Specific Modelling Languages

According to Zečević et al. (2017), languages can be classified into two categories: General Purpose Modelling Languages (GPMLs), which are applicable in every domain; and Domain Specific Modelling Languages (DSML), which are designed for a specific domain, context or industry.

A GPML provides rudimentary concepts such as “class”, “relation”, “attribute”. Frank (2010) defines a GPML as “a GPML is a modelling language that is thought to be independent from a particular domain of discourse. Instead, it should be suited to cover a wide range of domains. It consists of generic modelling concepts that do not include any specific aspects of a particular domain of discourse”.

In line with this definition, the Unified Modelling Language (UML) can be categorised as a GPML.

With the time, however, UML class diagram was deemed to be inappropriate when creating model, which was due its complexity and imprecise interpretation (France et al., 2006; Petre, 2013).

UML constructs do not directly address a specific problem domain. Thus, when modelling a reality within a specific application domain the so called “conceptual gap” between the problem domains and the solution domains is created (Mernik et al., 2005; France & Rumpe, 2007; Frank, 2010). Figure 9 helps to understand the conceptual gap. On the left-hand side of Figure 9 the abstraction from the underlying domain (i.e. the problem domain) is shown. On the right-hand side the reconstruction of an underlying information system, e.g. data structure, or software programs, (i.e. the solution domain) are depicted through rudimentary IT concepts such as “class”, “attribute”, “data type”, “state”, “state transition”.

When creating models from generic modelling constructs the general IT concepts are to be mapped to domain concepts, which consequently (according to Frank (2010)) raises unintended complexity. It can be compared to a business work written in primitive generic concepts only. Frank (2010) claims that this mapping induces a tremendous effort and is a threat to the model quality.

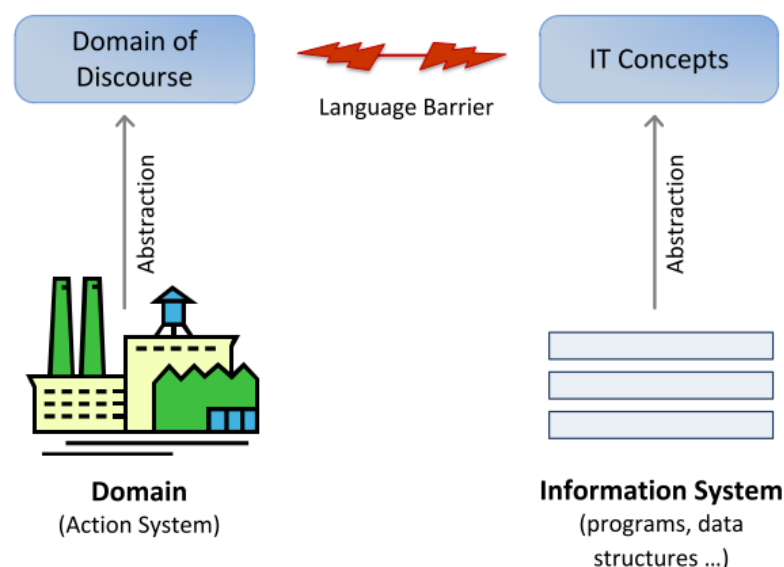


Figure 9. Language Barrier between Application Domain and Information Technology (Frank, 2010)

Another significant drawback of reconstructing a problem domain from general concepts is the high degree of freedom while modelling, afforded by GPML to the modeller. Such

freedom at modelling might lead to integrity problems, inconsistencies, unmeaningful representations and ambiguity of models. Thus, it makes it hard to understand the real meaning of models. Moreover, even if models are well designed, there is no way to enforce uniformity among them. The same reality can be modelled differently, and comparison among models becomes unfeasible.

In contrast to GPMLs, DSMLs (van Deursen et al., 2000; Frank, 2008; Clark et al., 2015(Frank, 2008)) are more expressive and concise (Zečević et al., 2017) as concepts and relations, and their notations are tailored to a specific problem domain. Tailored concepts and relations, together with constraints, are assimilated in to the meta-model. This provide the modeller with a less degree of freedom, which according to Kelly & Tolvanen (2008) allows designing in a less-error-prone manner. This supports the creation of consistent, meaningful and uniform models. As a consequence, productivity and integrity are fostered at inception (Mernik et al., 2005), leading to an increased quality of models (Frank 2013b).

Frank (2010) defines a DSML as “a DSML is a modelling language that is intended to be used in a certain domain of discourse. It enriches generic modelling concepts with concepts that were reconstructed from technical terms used in the respective domain of discourse. A DSML serves to create conceptual models of the domain, it is related to”.

The enrichment of generic modelling concepts with domain aspects takes place in the meta-model. UML class diagrams are often used to specify abstract syntax and constraints of DSMLs. As already mentioned, UML class diagram is used at level two of the meta-modelling hierarchy to specify modelling languages such as BPMN, CMMN, DMN and ArchiMate.

Enriching generic modelling concepts with domain aspects does not necessarily mean that more concepts or higher variety of concepts should be introduced in the meta-model. This mistake was the fate of many modelling languages like SADT (Marca & McGowan, 1988), or PSL (IEEE-SA Standards Board, 2005), where meta-models were enriched, inducing the side effect of increased complexity. Therefore, it is desirable to keep a lean meta-model with only necessary constructs and constraints. Consistently, van Deursen et al. (2000) define a DSML as a “a language that offers, through appropriate notations and abstractions, expressive power focused on, and usually restricted to, a particular problem domain”. This definition also intimates to the main benefit of DSML: the appropriate notations. Empirical studies conducted by Kosar et al. (2010) demonstrate a superior cognitive adequacy of domain-specific languages over general-purpose languages. The studies were conducted with respect to the 13 cognitive dimensions identified in Green and Petre (2008), namely:

- abstraction gradient, closeness of mapping, consistency, diffuseness, error-proneness, hard mental operations, hidden dependencies, premature commitment, progressive evaluation, role-expressiveness, secondary notation and escape from formalism, viscosity: resistance to local change, visibility and juxtaposability.

In a DSML, model users deal directly with language constructs that they are familiar with, leading to a significant positive impact on their learning curve. It was previously observed by Hudak and Paul (1996) that if domain experts can quickly learn the language its applicability improves. In turn, models that resemble an underlying reality are better understood among domain experts, which contributes to support optimisation phases. Pain points are rapidly identified, and actions can be taken accordingly. Ultimately decision-making is enhanced.

Domain experts performing modelling themselves is not just a trend. In some fields it has already become a reality. In enterprises, stakeholders use and design models to achieve common human interpretation so that decisions can be made, e.g. Business-IT alignment decisions. In the healthcare domain, modelling languages are also already widely used by physicians (or alternatively by quality managers), especially in process modelling as demonstrated by Braun et al. (2015).

A Domain-Specific Modelling Language in this work is defined as a graphical language that offers expressive power focused on a particular problem domain, through cognitive adequate notations and abstractions for humans. A DSML in this context serves to visualise, specify, construct and document aspects of an enterprise.

The notion “domain” does not have a fixed definition as it might refer to a paradigm, a business sector, an application area or a single case in an enterprise (Karagiannis et al., 2016). Frank (2010) states that it has never been used consistently in conceptual modelling. Also the difference between “specific-domain”, “crossed-domain” and “general purpose” is not clearly defined. A domain can, however, be less narrow - or narrower than others - and this determines the difference between languages that are less or more domain-specific, respectively. Karagiannis et al. (2016) introduce the notion of the *domain-specificity degree*, where a higher specificity degree means assimilating concepts in the metamodel that target a more specific domain.

For example, the modelling language BPMN embeds aspects for process modelling and it targets a more specific domain compared to UML class diagram. This higher domain-specificity degree or domain dependency is enough to categorise BPMN as a DSML. BPMN is more domain-specific than flow charts, which also contain activities, but has additional concepts for events, message flow and different types of tasks.

2.5.1 DSML vs GPML: A Practical Example

Figure 10 shows an excerpt of an IT infrastructure conceptualisation adapted from (Frank 2013b). The figure shows a comparison on level two (M_2) and level one (M_1) of how concepts are conceptualised in a DSML and in a GPML. Level two of the DSML contains two specific classes “Server” and “ERP” that are linked with each other with the specific relation “runs on” and contain attributes and attribute types. The instantiations of the three objects is straightforward and each are represented by appropriate graphical notations at level one. In this sense, level two already contains domain-specific knowledge. On the contrary, on the right-hand side of Figure 10, level two contains two general objects that are then classified in level one.

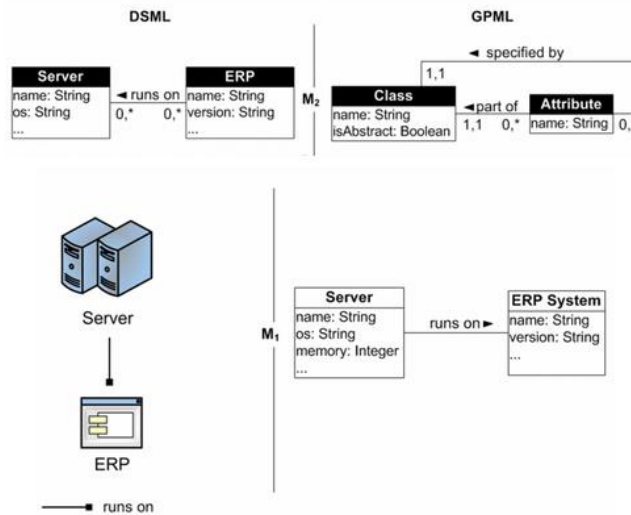


Figure 10. DSMLs vs. GPMLs adapted from (Frank, 2013b)

2.5.2 Terminology Definitions

In literature, the acronym DSML is sometimes interchanged with DSL, which is the acronym for *Domain-Specific Language* (Gray et al., 2008). However, the DSL mostly refers to *domain-specific programming languages* (DSPLs) (e.g. Ranabahu et al. (2012)), which differ from DSMLs as they are developed as extensions for dynamic programming languages and not in a model-based context. DSPLs have been widely adopted in software engineering, and like in DSMLs, they allow solutions to be expressed in the idiom and at the level of abstraction of the problem domain. According to van Deursen et al. (2000), a DSPL is defined as “[...] a programming language or executable specification language that offer, through appropriate notations and abstractions, expressive power focused on, and usually restricted to, a particular problem domain”.

DSPLs can be seen as both programming languages and specification languages. As such, they need dedicated tools with the capability to compile the (DSPL) program in order to function. The main purpose is to generate applications.

Research on DSPLs is more extensive and less recent than the research on DSMLs, which is relatively young. Domain-specific programming languages originated in Computer Science within the software engineering paradigm. Old programming languages like COBOL, FORTRAN, Lisp were already solving problems in certain areas, i.e. business processing, numeric computation and symbolic processing, respectively (van Deursen et al., 2000). van Deursen et al. (2000) introduce a wide list of key publications in the area of domain-specific programming languages and group them in different areas, i.e. financial products, behaviour control and coordination, software architectures and databases in the area of software engineering; simulation, mobile agents, robot control, solving partial differential equations and digital hardware design.

DSPLs are also called *micro-languages* and *little languages* (van Deursen et al., 2000). Hudak and Paul (1996) coined the name *domain-specific embedded languages* to refer to the result of extension syntactic mechanisms, such as definitions of functions or operators, from a base language. Extension mechanisms are common on model-based approaches as described in Sub-section 2.8.1.

DSMLs, in contrast to DSPLs, embrace the model-based paradigm. In the software engineering field, DSMLs fit to the above introduced MDE methodology, where models are used for the generation of executable code (Kelly & Tolvanen, 2008). Aquino et al. (2011) refer to it as the “modelling is programming” paradigm.

On the other hand, there are DSMLs that are instead employed with the purpose of creating knowledge bases to support decision making. These were created within the context of Enterprise Modelling (EM) that goes beyond the software development and aims at an holistic representation of the enterprise. EM spans research fields like Business Process Management (BPM), Enterprise Architecture Management (EAM), and, Knowledge Management (KM) (Karagiannis, 2018).

Both DSMLs for software development and those for the creation of knowledge bases share the same benefits of higher expressivity, conciseness, productivity, quality and direct involvement of domain experts (as introduced in Section 2.5). DSMLs for software development are subject to a series of model transformations (Avazpour et al., 2015; Jouault & Bézivin, 2006) until a code is generated. This approach demands for the specification of behavioural semantics of the language as it defines how the language elements can interact at runtime. In particular, the purpose of this specification is to dictate how the mapping of platform-independent language specifications should take place with respect to a specific software platform (Strembeck & Zdun, 2009). The behaviour can be specified in many ways, e.g. as a high-level control flow or over detailed behavioural models.

On the contrary, when the purpose of DSMLs is to retain knowledge like in enterprise modelling, the behavioural semantics are not considered.

As introduced in 2.2, in this work the focus is on DSMLs that retain knowledge and only structural semantics is considered.

2.6 Application Areas for DSML

This section presents some relevant DSMLs that were developed in different application areas. The DSMLs are grouped by their purpose: software development and knowledge retention.

2.6.1 DSML for Software Development

Frantz et al. (2011) presented a proposal called Guaranà with the aim to increase the level of abstraction of Enterprise Service Bus (ESB) as well as making it easier for software engineers to create, implement and deploy their Enterprise Application Integration (EAI) solutions. Guaranà provides explicit support to devise EAI solutions using enterprise integration patterns by means of a graphical notation. The DSML offers a view of the whole set of processes of which EAI solutions are composed, enabling software engineers to devise EAI solutions at a high-level of abstraction. Furthermore, a graphical tool editor was implemented with a set of scripts to transform the models into Java code ready to be compiled and executed.

Nunes and Schwabe (2006) describe their “HyperDe” system as an environment to support the rapid prototyping of web applications with the use of DSMLs. The latter allows developers to create code by manipulating models that specify the application. The HyperDe environment supports designing web application through a meta-model instantiation. Furthermore, the HyperDe extends the Ruby on Rails (RoR) framework into a domain-specific language, allowing direct manipulation of both the model and the meta-model within Ruby scripts.

Similarly, Cadavid et al. (2009) developed a DSML that aimed at simplifying the web application development through models. The work describes the elements used in the process of transforming a UML domain model into a deployable web application. In this way they demonstrate that models can be transformed and executed for the automatic generation of applications.

Zhou et al. (2011) propose a language, called WL4EA (Web Language for Enterprise Application) for interactive development of Enterprise Applications (EAs). In this case, the model-driven approach was adopted to lower technology complexity and improve productivity. Meta-models were used to specify the WL4EA and contain the three layers: user interface, business logic and data persistence.

All the presented approaches allow for an automatic generation of applications and claim to significantly accelerate the development process. Whereas the first two address the same domain (i.e. general web applications), the latter targets the more specific domain of enterprise applications. The main difference between the first two approaches is, however, the technology adopted to generate applications, i.e. Eclipse platform by Cadavid et al. (2009) and the web development framework Ruby on Rails. Ruby on Rails implied an extension of the programming language from which a DSPL was developed.

2.6.2 DSML for Knowledge Retention

DSMLs that are created for knowledge retention are used either only to document enterprise knowledge or for knowledge automation (Karagiannis & Woitsch, 2015; Hinkelmann et al., 2018). Knowledge documentation is practiced in the field of Knowledge Management while automation of knowledge is a common practice in the field of Knowledge Engineering (Karagiannis & Woitsch, 2015).

As mentioned in (Karagiannis & Woitsch, 2015; Hinkelmann, Gerber, et al., 2016; Hinkelmann et al., 2018) *Knowledge Engineering* (KE) focuses on the machine interpretation of knowledge (see Section 2.12), whilst *Knowledge Management* (KM) focuses on the human interpretation of the knowledge. The interpretation is one of the four dimensions of the

knowledge space depicted in Figure 11. According to Karagiannis and Woitsch, (2015) the term knowledge space refers to what is represented in a model. The other three dimensions are the *form*, *content* and *use*. The form reflects to specification in terms of syntax and semantics. The content is the actual knowledge contained in the model, which abstracts from a certain problem domain. Finally, the use refers to the part of the content that is used for a particular purpose.

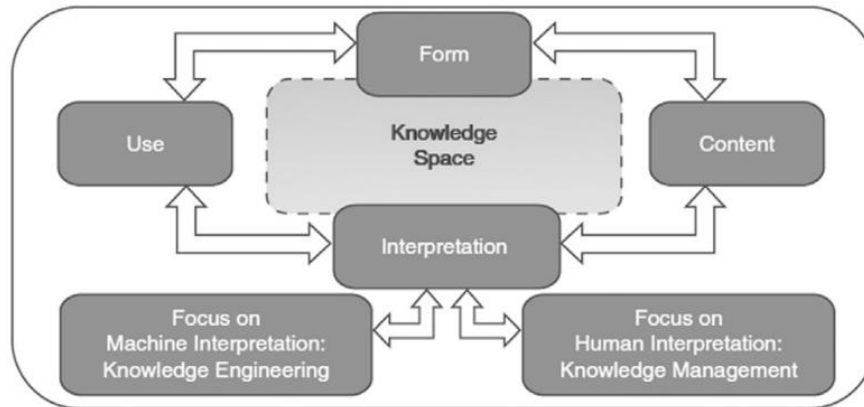


Figure 11. The four dimensions of a knowledge space (Karagiannis & Woitsch, 2015)

There has been an increased development of DSMLs for knowledge retention over the past few years. For instance, Karagiannis et al. (2016) contains a wide list of DSMLs, some of which are used only for knowledge documentation while others are also used for knowledge automation. The ultimate goal is to support decision making.

For instance, the Fundamental Conceptual Modelling Language (FCML) (Karagiannis et al., 2016) integrates and extends concepts from five different modelling languages (i.e. Entity-Relationship, Event-driven Process Chain, BPMN, UML, Petri Nets) addressing the domains of business process management and software engineering. FCML enables a multi-purpose modelling in the same tool. Concepts contained in FCML are extended with domain-specific properties and hyperlinks to connect models with each other. This provides the appropriate syntax and semantics for process-based simulations, based on which, process improvement decisions can be made.

Another DSML is the Business Process Feature Model (Cognini et al., 2016), which supports variability for Business Process modelling. The authors developed this modelling notation from the need to represent variability in the domain of Public Administration (PA). PA offices from different institutions have similar rules to comply with when providing services to citizens. However, different offices have different characteristics affecting the way a service is delivered. This most often leads to internal rules to be adapted the characteristics of each different office. For this, the BPMN meta-model was adapted so that activities forming a configurable business process model could be successively refined to consider specific characteristics of a deployment context.

The Knowledge Work Designer (Hinkelmann, 2016a) is a modelling tool, which combines modelling languages for both structured and non-structured process logic and business logic as well as a new integration of business process and case modelling. BPMN and CMMN are deeply integrated resulting in a new modelling language BPCM. With respect to modelling business logic, the modelling language spans, on one side, the standard Decision Model and Notation (DMN) to model structured decision logic. On the other side, the language spans the Document and Knowledge modelling language (adopted also by De Angelis et al. (2016)) to

model unstructured decision logic such as guidelines, checklist, sample outputs, or templates. Figure 12 depicts the mentioned modelling languages integrated with each other.

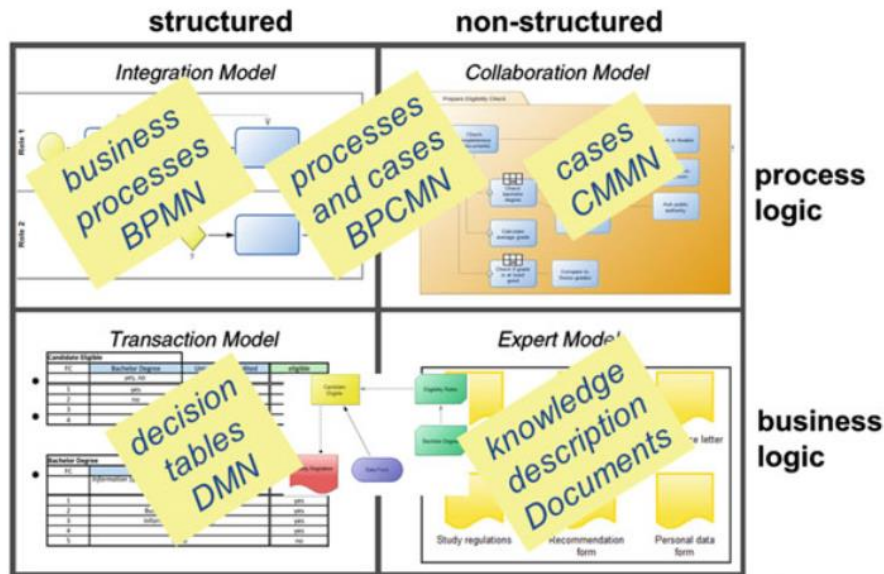


Figure 12. Integrated modelling languages in the Knowledge Work Designer modelling tool. Adapted from (Hinkelmann, 2016)

The new modelling language was applied to the admission process of the Master of Science study programs at School of Business FHNW (Hinkelmann, 2016a).

In Emmenegger et al. (2017) and in De Angelis et al. (2016), modelling languages are adapted and integrated to create models used to support decision-making of unexperienced employees in workplaces. Namely, several recommendations are provided for which certain knowledge needed to be made explicit. For example, to recommend experts like (a) line managers, (b) more experienced employees covering the same role in other organisations, and (c) people who executed a specific task most often, the following knowledge was modelled:

- Work context of the learner: current executed task and the business process. For this BPMN was adopted.
- Work experience of the learner. For this, logs of processes and tasks instances of each performer are analysed.
- The role of the learner and his/her position in the organisation. For this, BPMN was integrated with the Organisational modelling language where tasks are related to employees and lanes are related to the roles performed by each employee.
- Organisational structure, which was modelled by the Organisational model.

Similarly, learning materials like links, video files or simulation are derived from the knowledge modelled by integrating several modelling languages. Namely, the Document (Data) modelling language was used to model all possible acquired competency levels of an employee and the required competency level to fulfil a role. Hence, the modelling language is integrated with the Organisation modelling language. A higher required company level than the level acquired implies that some learning materials should be recommended. To specify the competency levels a new modelling language was developed, which is based on The European Qualifications Framework (EQF). Thus, an integration between the Document (Data) meta-model and the Competency meta-model was necessary. The Competency meta-model allows for categorizing of the competency levels in different learning outcomes. In turn, the learning

outcomes are related to learning goals that are specified in the Business Motivation Model (OMG, 2014a).

The use and development of a DSML involve several roles. The following section aims to clarify the distinction between the different roles around a DSML.

2.7 Roles in DSML

In keeping with Karagiannis and Kühn (2002), Kleppe (2009) and Barišić et al. (2018), the *language engineer* or the *DSML developer* is skilled in the engineering discipline of developing modelling languages. This role is responsible for an adequate definition of the syntax, semantics, and notation. Within the engineering life-cycle of a DSML (see Section 2.9), this role is involved in the modelling language specification, implementation and evaluation.

In DSMLs that are used for knowledge retention (see Section 2.6.2), the language engineer is also considered to be skilled in both KE and KM. Namely, expertise in KE enables the language engineer to create knowledge bases or models that are used and interpreted by machines, thus their knowledge representation is required to be formally structured like in ontologies (see ontologies in Section 2.13.1). Expertise on KM, on the other hand, refers to the ability of creating models used and interpreted by humans, thus their knowledge representation is required to be cognitively adequate. Both KE and KM are typical knowledge approaches that use models to conceptualise an underlying reality. This concept is depicted in Figure 13, where models for human-interpretability are typically separate from those for machine-interpretability.

Thus, with respect to the definition of a modelling language introduced in Section 2.2, KE and KM enable a language engineer to specify abstract syntax, constraints, additional semantics and graphical notations so that models created with these can be interpreted by both human and machines.

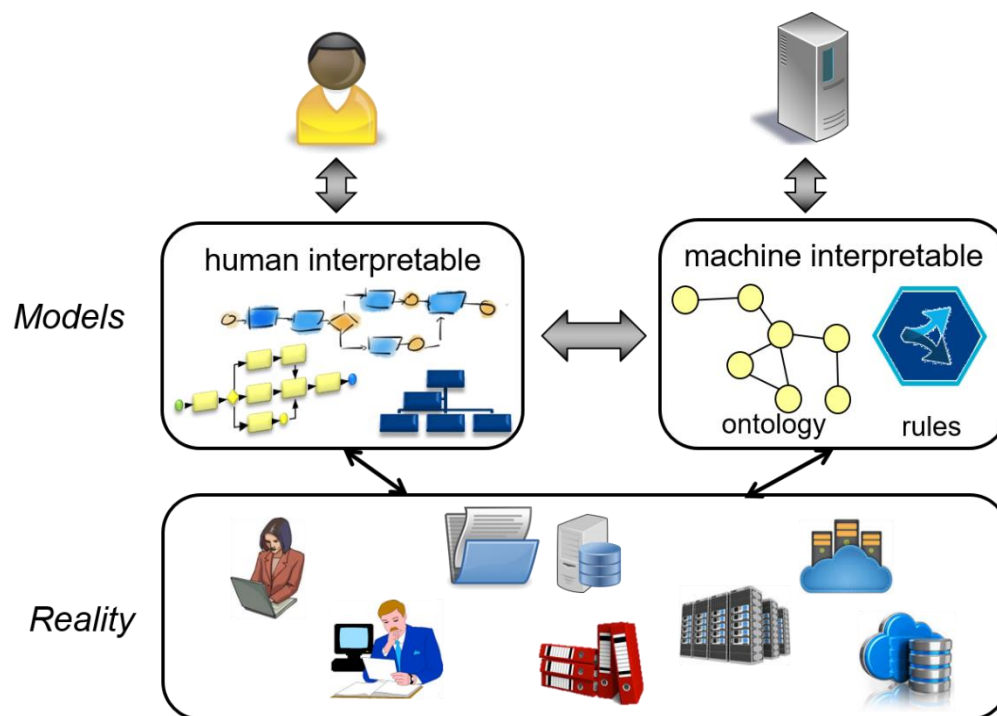


Figure 13. Typical modelling approach: separating models for human and machine interpretation (Hinkelmann et al., 2018)

The *modelling tool engineer* is any person skilled in the development of modelling tools. This role is responsible for the implementation of a DSML in a correspondent tool. Hence, capabilities and procedures pertinent to a modelling language should adhere to the modelling tool in which the model is used. According to Karagiannis and Kühn (2002), this role has the ability to configure mechanisms of a meta-modelling platform for particular meta-models. The language engineer is most likely to also have expertise in modelling tool engineering.

The *modelling expert or DSML user* is any person who is skilled in modelling and is applying one or more modelling languages. The modelling expert uses modelling languages to create models and might provide guidance or explanations to domain experts. He or she is should be involved in the evaluation of a modelling language to provide feedback, hence they will also be in direct contact with the language engineer. Karagiannis and Kühn (2002) refer to this role as the process engineer, while in industry it is likely to be called a modelling (senior) consultant. The modelling expert works at level one if referring to the meta-modelling hierarchy introduced in Section 2.3.

According to Kelly and Tolvanen (2008), in software engineering the modelling expert is in charge of creating applications that relate to specific characteristics of configurations like specifying deployment of software units to hardware. Thus, human interpretation plays a minor role as the aim of these models is to be read by machines.

Conversely, in enterprise modelling, the modelling expert creates models either for the human interpretation (e.g. for enterprise architecture stakeholders), for the machine interpretation (e.g. automation of processes or knowledge inferencing), or both.

The *domain expert* is any person who is skilled in a specific domain, which is targeted by a DSML. He or she can cooperate with the modelling expert to create models pertaining to the underlying domain. Evaluation of the modelling language is most likely to be performed directly between the language engineer(s) and the domain expert(s), with no involvement of modelling experts (Völter et al., 2013). In some case this role has the ability to create models without assistance of the modelling expert. In other cases, a domain expert is just requested to understand or use a model, or simply interact with a model instantiation (e.g. interaction of end-users with process instances in Business Process Management). The domain expert operates at level one or zero of the meta-modelling hierarchy.

In software engineering, the domain expert does not need to have the software development background, but has the ability to specify the application for code generation (Barišić et al., 2018).

In enterprise modelling, the domain expert relates to the human interpretation of models only. Thus, ideally, modelling constructs of a DSML have cognitively adequate graphical notations, which can easily be understood by a domain expert (Hinkelmann et al., 2018). Karagiannis and Kühn (2002) refer to this role as the *method user*, who creates models by using the modelling language, following the modelling procedure and applying the available mechanisms. According to the above definition, a method user can also be a modelling expert.

One can distinguish between two approaches of DSML engineering. The following section introduces both and motivates the focus on one over the other in this thesis.

2.8 Approaches for DSML Engineering

Engineering a DSML can be done “from scratch” by building dedicated DSMLs with a complete new set of modelling constructs targeting a given domain. One example is the work of Burwitz et al. (2013), where the CPmod modelling language was developed to model clinical pathways. In this case, benefits of a DSML like higher expressivity of the language and thus higher productivity at design time, increased quality of the final product. However, the direct inclusion of the final user to the engineering process resulted in high development costs (Mernik et al., 2005). These costs include the designing, the implementation and maintenance of both a DSML and a corresponding modelling tool, from which the language can be used.

Frank (2013a) claims that these high costs are difficult to justify beforehand as they are highly dependent on how well the language and the corresponding tool are designed, implemented and maintained. For instance, bad decisions made during the development phase of a DSML (see Section 2.9 for more challenges) leads to a poorly designed DSML affecting the tool upon which the language is implemented.

It might also be the case, however, that a well-designed DSML is limited by the corresponding tool. If the tool is not appropriately developed, on the one hand, it can limit the capabilities of the language (Gray et al., 2008); and on the other hand, it can hinder the model exchange across systems (Braun 2015b).

These issues make the benefits of a DSML rather difficult to predict, e.g. high quality of models and increased decision support. In turn, based on the experience of Frank (2013a), managers are difficult to persuade when it comes to initial investment in DSMLs.

An additional drawback of dedicated DSMLs regards the limited reuse of language concepts. Dedicated DSMLs prevent the reusability of the language concepts in at least the following three scenarios:

- in projects where the domain-specificity degree is different, e.g. if a project requires to abstract away from a DSML or bringing more details to it;
- in projects that require the same domain-specificity degree but address new stakeholders. Different stakeholders most likely have a different understanding or a different perception of the underlying reality, of simply different requirements. This leads to a different conceptualisation of the language;
- within the modelling community as syntax and semantics of the DSML would only be known by those stakeholders involved in the project.

A solution to remedy the presented drawbacks of dedicated DSMLs is to rely on *domain-specific adaption* approaches, which are performed on existing modelling languages. A domain-specific modelling approach promises to significantly decrease the development costs of a DSML (Chiprianov et al., 2013).

In the next sub-sections, the approach for domain-specific adaptations is expanded together with the existing different techniques. A comparison among the different techniques is provided, which leads to motivate the choice of one technique for this research work.

2.8.1 Advantages of Domain-Specific Adaptation

Existing modelling languages address a wide class of stakeholders and allow modelling patterns and invariants that are recurring across application areas. BPMN, for example, is used to model business processes in many different business sectors. Developing a DSML by adapting modelling languages provides the benefit of considering established experience and lessons learned and notations from these languages (Karagiannis et al., 2016). Particularly,

modelling standards come with sets of proven and well-known concepts with a clear syntax and a widely accepted semantics. In turn, DSMLs remain more intuitive for modelling experts. As claimed in (Braun, 2015b), this approach also avoids to create yet another modelling language, which possibly has redundant overlaps with basal concepts from standard enterprise modelling languages. In the same line, Robert et al., (2009) motivates the adaptation approach based on the fact that a lot of DSMLs end up having shared equivalent concepts and representations.

Jablonski et al. (2008), specify the modelling language adaptation as an “extension or extensibility so that domain specific requirements can be integrated or domain specific semantics are better reflected”.

Atkinson et al. (2013) consider the extensibility of modelling languages extremely relevant in the enterprise modelling domain, especially to the vast amount of stakeholders in relation to their perspectives on the enterprise. Also, extensibility comprises the opportunity of improving and evolving an enterprise modelling language (Braun, 2015b). Recurring extensions can act, for instance, as evidence for changing a standard language. For example, this was the case of BPMN that was subject to many improvements from version 1.0 to version 2.0, e.g. the choreography extension was integrated in BPMN 2.0 (Decker et al., 2007).

Braun et al. (2015) built a DSML through an extension of BPMN. They provide insights on the approach adopted by comparing it to their previous work in Burwitz et al. (2013), in which a dedicated DSML was built from scratch addressing the same application domain, i.e. clinical pathways in healthcare. The comparison includes eleven criteria. The criteria and the comparison results are reported in Figure 14, where a filled dots means fulfilled criteria, an empty dot means partly fulfilled criteria, and “-“ means not fulfilled criteria).

Criterion	Extension Approach		DSML Approach (see [5])	
<i>Domain Representation</i>	Nearly complete * Discussion on conditional equivalences necessary (see Table 1)	● (*)	Complete	●
<i>Meta Model</i>	Level M2 / M2 _{profile}	●	Level M2	●
<i>Abstr. Syntax</i>	Limited capabilities (missing element constraints), no views possible	⊙	Complete specification Views are possible	●
<i>Concr. Syntax</i>	Limited by BPMN style ([24], p. 8)	⊙	Not limited	●
<i>Procedure</i>	Limited guidance [1, 36]	⊙	Limited guidance [10]	⊙
<i>Concept Reuse</i>	Broad reuse (e.g., task type) Focus on domain concepts	●	Re-design of basal process concepts Focus on domain and process concepts	-
<i>Cost of Design</i>	Depends on relations to BPMN	⊙	Depends on language	⊙
<i>Tool Support</i>	Good (range of tools)	●	Limited	-
<i>Dissemination</i>	Very good	●	Limited	-
<i>Integration</i>	Integration with other extensions Reuse of other BPMN interfaces	●	Dedicated interfaces required No integration reuse	-
<i>Execution</i>	BPEL integration with adaption	⊙	Requires dedicated design	-

Figure 14. Comparison between "domain-specific adaptation" of BPMN and "from scratch" approach to develop DSMLs (Braun et al., 2015)

From the comparison results, one can observe that a domain-specific adaptation approach presents more advantages in terms of reusability of concepts, tool support, dissemination, integration, and execution.

A further example of domain-specific adaptation is provided in Chiprianov et al., (2013). The authors extended the enterprise modelling language ArchiMate 2.1 (The Open Group, 2012b) to model the “Design Rationale” (DR), which is the justification behind decisions regarding the design of an artefact. Figure 15 shows the language extension applied on the meta-model of ArchiMate. The new concepts are defined as sub-concept of “ArchiMate

Element”, which is a root concept in the ArchiMate meta-model made available to specify extensions. Figure 16 shows the DR DSML as a result of such an extension.

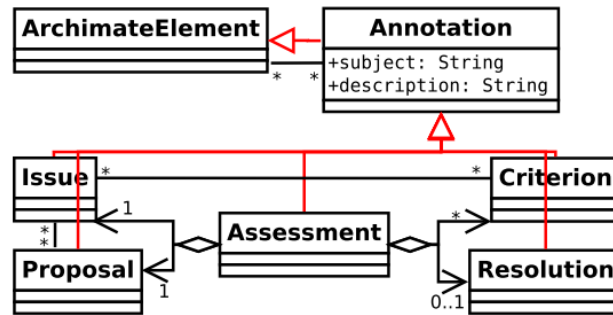


Figure 15. The abstract syntax of DR DSML (Chiprianov et al., 2013)



Figure 16. The concrete syntax of the DR DSML (Chiprianov et al., 2013)

In both research works, extensions for BPMN (Braun et al., 2015) and for ArchiMate (Chiprianov et al., 2013) a *profiling mechanism* is utilised as a domain-specific adaptation technique. In the following sub-section, the common domain-specific adaptation techniques are described.

2.8.2 Strengths and Weaknesses of Common Techniques for Domain-Specific Adaptations

Multiple different types of techniques exist for domain-specific adaptations of modelling languages. Atkinson et al. (2015) distinguish among three main techniques: *In-built*, *Model Annotation*, and *Meta-models Customisation*. Braun, (2015b) builds upon this distinction to further elaborate on them. The three technique types are described below, along with their related strengths and weaknesses.

In-built techniques are also called *integrated extensibility mechanisms* as they inherent mechanism for language extension from the meta-level (M_2). These mechanisms allow to customise the meta-model without changing it. A widely used in-built extension mechanism is the *profiling mechanism*. Here, well-defined language elements residing in M_2 can be applied for the specification of new elements on the model level (M_1). Specifications of some standard

languages like UML 2.5.1 (OMG, 2017), Business Process Management and Notation 2.0 (OMG, 2011), and ArchiMate 3.0 (The Open Group, 2017) foresee these special “built-in” mechanisms. These should eventually be implemented in a modelling tool. As an example, the tool Visual Paradigm (Visual Paradigm, 2018) implements extensibility mechanisms (like *stereotype* and *tagged values*) foreseen by ArchiMate. Such in-built mechanisms, however, are criticised by many authors within the practice of enterprise modelling as they suffer from several problems. Some of these problems are listed in the following list:

1. *Limited to language extensions*: the core classes in the meta-model cannot be changed (Atkinson *et al.*, 2013).
2. *Abstraction conflicts*: the extension occurs in the meta-model by instantiating the core classes, thus creating a new version of the meta-model with a lower degree of abstraction. This goes in contrast with the definition of the MOF’s four-layer architecture (OMG 2016c) as an extra layer would be created between the model and meta-model layer (Braun, 2015a) (see the four-layer architecture in Section 2.3);
3. *Limited specification of extensible concepts and language adequacy issues*: profiling mechanisms (the most known and used in-built extensible mechanism) originated to extend classes in UML, wherein the general class type is extended to define new types. The generic class type is, however, not available in enterprise modelling languages that are more purpose- (or domain-) specific like BPMN (Braun, 2015a). Consequently, the definition of new types are hampered (Braun, 2015a). In BPMN, although generic concepts like Artefacts can be used for extension, adaptations result in limitations to expressiveness and adequacy of language (Braun, 2015a; Braun, 2016). Additionally, the limited specification of extensible concepts can also derive by the fact that mechanisms are not fully implemented by tools (Atkinson *et al.*, 2013).
4. *Additional complexity due to the limited expressiveness*: the limited expressiveness of the extensibility mechanism may lead to the need for the constraint of language such as OCL (OMG, 2014b), which can be problematic. Salehi *et al.*, (2016), for example, stress the need to not only use OCL but also extend them to the considered application domain, which was inconvenient.
5. *Domain-specific dependencies between extended elements are not always possible or unclear*: the possibility to establish relationships between extension concepts is rather unclear (Braun, 2015a). In BPMN, aggregations or directed relations between extension concepts is not considered (Braun, 2015a).
6. *Difficulties in handling extensibility mechanisms*: studies on the main OMG standard profiles conducted by Robert *et al.*, (2009) showed not only the uneven quality of the resulting language but also the severe difficulties in mastering such mechanisms due to inappropriate use.
7. *Lack of both a systematic approach and methodical guidance for the language extension* (Robert *et al.*, 2009; Braun, 2015a; Salehi *et al.*, 2016): this is problematic for the good language extension. Salehi *et al.*, (2016) investigated existing designed (standardised and non-standardised) UML profiles and found them to be technically invalid, in direct contradiction of the UML meta-model, or of poor quality.
8. *Problems of validation and consistency for models created with extended modelling languages*: these problems arise when the extensibility mechanisms do not allow the detailed specification of extended modelling elements (see also problem 3). For instance, this is the case of BPMN. In BPMN every extended class has to instantiate the

super class Base Element (Braun, 2015a). Only then it can be assigned to a particular definition;

9. *Compatibility and interoperability problems*: in-built extensibility mechanisms are not standardised across tools and modelling languages, which leads to significant compatibility and interoperability problems within the extended language (Atkinson et al., 2013).
10. *Exchangeability problems*: these problems raise due to the lack of detailed consideration in mechanisms regarding the exchangeability of both the abstract syntax and the concrete syntax of the extended elements (Braun, 2015a) (see Section 2.2 for definitions of concrete and abstract syntax);
11. *Mechanisms not applicable to all modelling languages*: as depicted in Figure 17, most of the reported modelling languages do not fully foresee extensibility mechanisms for both abstract and concrete syntaxes and neither have supportive procedures (Braun 2015b). Although others, like ArchiMate, consider such mechanisms, they do not provide a well-defined meta-model of all extension-relevant concepts nor do they present a procedure model for its application (Braun 2015b). For languages like CMMN and DMN there is no conceptual understanding of extensibility, but only forbidden extension operations (Braun 2015b).

		Syntax		Procedure
		Abstract	Concrete	
OMG languages	UML	●	⊙	⊙
	BPMN	●	⊙	⊙
	CMMN	–	–	–
	DMW	○	–	–
	SBVR	○	–	–
	KDML	○	–	–
	Essence	●	–	–
	IFML	⊙	–	–
	SysML	○	–	–
	KDM	●	⊙	⊙
	SMM	○	–	–
	⋮	ArchiMate	⊙	–
	ARIS	○	–	–

● (exists) ⊙ (partially) – (does not exist)

Figure 17. Extensibility of abstract and concrete syntax within MOF-based languages and their provision of methodical support for extension design (Braun, 2015b)

An alternative to the in-built approach is the *Model Annotation*. In this approach the language extensions are not defined as part of the language, but rather attached to the language as an external language. The attachment is achieved by the model weaving (Didonet Del Fabro & Valduriez, 2009), which stores the additional information needed by the extended language. The model annotation approach keeps the two languages (i.e. the language to be annotated and the one for annotation) completely separate. This implies the support of two separate tools, i.e. one for the annotated information and one for the target model (i.e. the model that is annotated). Atkinson et al. (2015) regard this approach as a valid one for modelling language *augmentation*. That is, the extension of the original language is performed with concepts, attributes and rules from a different domain (Braun (2015b) calls it *vertical extension*). For

instance, the enhancement of BPMN with data for performance simulation. This approach would also be an option in case language engineers are somehow forced to adopt a modelling tools that are either closed-source (i.e. meta-model hardwired and thus not accessible for changes), or do not implement “built-in” extension. For instance, the meta-model Ecore from the Eclipse Modelling Framework is used to implement a modelling tool as an Eclipse plugin, which has no built-in extension mechanism (Atkinson et al., 2015). The same approach is not advisable, however, when extending a language with elements from the same domain (e.g. a new gateway type is introduced within BPMN), i.e. language *enhancement* (Atkinson et al., 2015), or *horizontal extension* according to Braun (2015b). The drawback is that unnecessary duplicated model elements are often created, i.e. same elements appear in two languages.

A further alternative to these two approaches is the *Meta-models Customisation* (also known as *ad-hoc customisation* of meta-models). This approach performs a straightforward language adaptation (both language augmentation and enhancement) by directly changing the language’s meta-model. This approach avoids the drawbacks presented for both in-built and model annotation approaches, and (according to Atkinson et al. (2013)) it often leads to extension definitions that are better structured and to a higher quality in terms of system engineering maxims such as “separation of concerns”, “high cohesion” and “low coupling”. Notably this approach does not hinder the integration of concepts from different modelling languages. Conversely, this is the case of built-in mechanisms (see above drawbacks (d) and (e)). In a systematic literature review focusing on BPMN extensions, Braun and Esswein (2014) found that most of the extensions (approx. 80%) do not use the built-in approaches, but rather the meta-model customisation.

However, the meta-model customisation lacks the methodological support for the meta-model customisation (Braun et al., 2015) and shares some of the challenges afflicting the design of a dedicated DSML (see Section 2.9). Hence, the correct customisation remains up to the language engineer. Some practical issues that might obstacle the adoption of this approach pertain to the chosen modelling tool. As stressed by Atkinson et al., (2013), many frameworks and tools have either an hardwired meta-model, which are not accessible for changes, or extensions are made possible but cannot be used in the modelling tool. The latter requires the modelling tool to be re-compiled and re-deployed to implement the performed extensions.

Braun (2015b) classifies the Multi-Level Modelling (introduced in Atkinson and Kühne, (2003), Atkinson and Kühne (2008), Atkinson et al. (2013)) as a further and novel extension mechanism. The approach proposes a flexible multi-level architecture based on two completely orthogonal dimensions of classification - one dealing with linguistic classification and the other dealing with domain (i.e. ontological) classification (Atkinson & Kuhne, 2003). The approach addresses the abstraction limitation caused by the dichotomy of “class” and “instance”, which occurs when changing the language level (for a detailed understanding see the concept of “clabject” described by Atkinson and Kühne (2008)). The authors stress the fact that this limitation is carried by the rigid four-layered OMG architecture. However, this approach is not yet widely diffused, especially in industry (Atkinson & Kühne, 2017). According to Atkinson et al. (2015), one of the key shortcoming in this approach is the support of the above-defined language augmentation, as it needs the dedicated modelling tool MelanEE to be implemented. The latter induces to the drawbacks of designing a DSML from scratch introduced in Section 2.8.

Similarly, Frank (2011; 2014) proposes a multi-perspective enterprise modelling (MEMO) and a correspondent modelling environment. The author uses a three level modelling hierarchy for conceptualisation purposes and, similar to Atkinson et al. (2015), addresses the abstraction issue caused by the dichotomy of “class” and “instance”. According to Frank (2014), his approach is designed to be far easier to use than Atkinson and Kuhne's approach (2003). The approach is limited to the linguistic view only, and suffers from similar drawbacks to the

previous approach, i.e. scarce diffusion in industry and language implementation that requires a dedicated tool.

2.8.3 Reflections and Considerations

This research work aims to conceive an agile approach for the quick delivery of high quality DSMLs. It is therefore, fundamental, to identify an appropriate approach supporting this aim. For this, an in-depth literature investigation was performed starting from the prominent engineering approaches for DSMLs. Findings revealed the significant benefits of domain-specific adaptations of modelling languages over dedicated DSMLs. Therefore, the investigation focused on techniques for domain-specific adaptations, from which strengths and weaknesses were highlighted. In the following, reflections and considerations are provided to motivate the choice of one domain-specific adaptation technique supporting the aforementioned aim of this research work.

There are two major problems with *built-in* and *model annotation* techniques that prevent the quick engineering of DSMLs and hamper the quality of the resulting DSMLs: the increased complexity in the practice of the language adaptation and the introduction of redundant concepts, respectively (Atkinson et al., 2015) - see Sub-section 2.8.2. Another limitation afflicting both the built-in and model annotation techniques, is about their restriction to language extension. That is, the core meta-model of a modelling language cannot be changed. Such a limitation contrasts with agile approaches wherein a high degree of flexibility in language adaptation is required.

These limitations are enough to conclude that both the language adaptation techniques *built-in* and *model annotation* are not conducive to the agile meta-modelling approach that is targeted by this work.

The two meta-modelling approaches proposed in (Frank, 2014; Atkinson & Kühne, 2008), address common abstraction issues carried by approaches that follow the four-layered OMG architecture. While the abstraction issue is an important point to consider in meta-modelling, one of the major problems of the two approaches is that they are not intuitive for practical use. For example, as stated in (Atkinson et al., 2015) extra effort would be needed to implement workarounds to address the language augmentation issue (see language augmentation in Sub-section 2.8.2). For this reason the two approaches of Frank (2014) and Atkinson and Kühne, (2008) are not considered, because the agile approach targeted by this research work aims to be relevant for the practice. The abstraction issue in this research work is handled as follows:

The abstraction issue is, ultimately, strictly related to the kind of reality that needs to be conceptualised. It might be that for a given application domain, the conceptualisation over two levels (model and meta-model) is enough, like the work in (Reimer & Laurenzi, 2014). Sticking to the required levels, avoids adding unnecessary complexity when creating models. Refined capabilities can then be added only if required by the application domain. In keeping with this argument, in this research work the kind of reality that needed to be conceptualized in the two addressed application domains (see Sections 0 and 4.2) determined the level of abstraction. From the two application domains, the conceptualisation over two levels (model and meta-model) was enough.

The *meta-model customisation* (or *ad-hoc customisation*) seems to be the most appropriate technique to support the desired agile approach. In contrast to the built-in and annotation techniques, it offers a high degree of flexibility in the language adaptation. That is, it allows

not only to extend but also to adapt the meta-model. Moreover, it does not request to follow particular rules to overcome abstraction issues while adapting the language (like in the built-in techniques). It trades *simplicity of use* for *abstraction limitations* when compared to the multi-perspective modelling of Frank (2014) and multi-level modelling of Atkinson et al. (2015). However, as already described, the abstraction limitations are not a problem in this research work. The simplicity of use is preferred as it supports agility in the quick delivery of DSMLs and models.

The *meta-model customisation* approach also benefits from the advantages of dedicated DSMLs (i.e. higher expressivity, conciseness, productivity, quality and better cognitive adequacy of notations - see Section 2.5) together with the advantages of considering existing languages (i.e. experience and lessons learned from existing languages, reusability of well-known and establishes set of concepts, increasing integration with recognised standards). Lastly, the *meta-model customisation* is a widespread approach in both research and industry to adapt modelling languages, which increases the confidence of its adoption.

The notion “*domain-specific adaptation*” in this research work refers to the *meta-model customisation* approach (also known as ad-hoc customisation approach). Domain-specific adaptations, thereby, not only allow for extending an existing modelling language but also to change it and integrate it with other modelling languages.

Engineering a DSML through domain-specific adaptations of modelling languages is still in its infancy. Braun (2016) and Braun et al. (2015) claim there are few research articles explicitly addressing the topic of extending a modelling language that evokes alteration of the underlying meta-model. An in-depth investigation on the challenges related to such an approach was consequently conducted. The main challenges from the findings are consolidated in the next section.

2.9 DSML Engineering: Main Challenges

Research concerning problems related to the domain-specific adaptations (or meta-model customisation) for DSML engineering has shown that the design of the meta-model is the most critical challenge. A summary of these findings is given below.

The complexity of when designing conceptual models, which in GPMLs resides in level one, in DSMLs shifts to level two, i.e. to the meta-model (Mernik et al. 2005; Gray et al. 2008; Fowler 2011). That is, abstract syntax, constraints need to be specified in the meta-model. Bork and Fill (2014) stress the scarce availability of guidelines and best practices. Consistently, Frank (2013a) asserts that there is still lack of methods to guide the designing phase. As Cho et al. (2012) highlight, the design of a meta-model remains a crucial task even for language engineers with high expertise. It requires both language development expertise and domain knowledge, and few people rarely have both (Mernik et al., 2005; Cho et al. 2012; Chiprianov et al., 2013). The domain knowledge mostly resides in users or domain experts’ minds. Therefore, it is fundamental for the language engineer to cooperate with them to extract and make explicit the needed knowledge (Izquierdo et al., 2013; Barišić et al., 2018). Izquierdo et al. (2013) and Barišić et al. (2018) claim that the lack of cooperation with end-users while developing a DSML is likely to cause misinterpretations, which hamper the development process and the quality of the DSML. This can also lead to problems on finding, setting and maintaining a suitable scope for the DSML (Frank 2010).

Moreover, language engineers face an array of challenges, from complex decisions relating to the appropriate abstraction level (Wegeler et al., 2013; Karagiannis et al., 2016) (or domain-specificity degree as above mentioned) of modelling constructs. Therefore, trade-off decisions between productivity and re-usability have to be made (Frank, 2013a) – for example, the more a language construct is reusable across domains, the lower the degree of domain-specificity (or level of semantic as defined by Frank (2010)) and thus the language construct is more generic (see Figure 18). In such a case, the modeller may need to conceptualise some more domain-specific concepts (at level one), which leaves room for broader interpretation. This, however, leads to a decrease in the productivity of a DSML and, consequently lowers the effective support of a DSML. Conversely, increasing the domain-specific degree of modelling constructs, increases the productivity level of the modelling language, and it lowers the possibility to reuse the DSML (or single language constructs) across domains or even in different areas, processes or projects of the same domain. Frank (2013b) refers to this trade-off as the conflict between the benefits and drawbacks of semantics, and it is considered as a fundamental challenge for information system design in general.

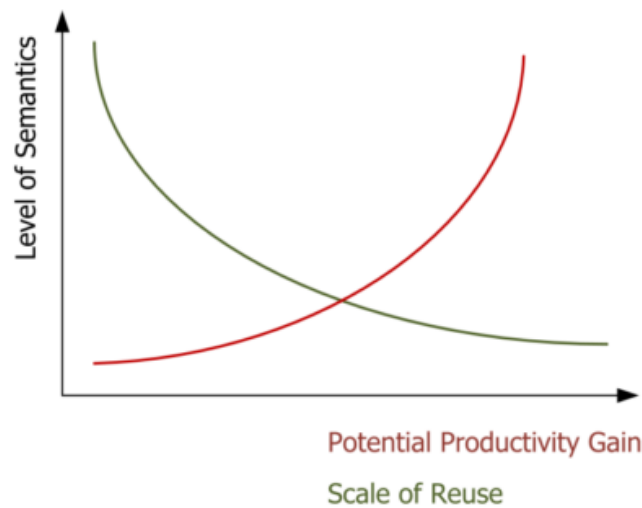


Figure 18. DSMLs: Potential Productivity Gain vs. Scale of Reuse (Frank, 2010)

In conclusion, involving the end-user in an early phase in the engineering lifecycle of a DSML promises to alleviate the above-mentioned pain-points. This strategy leads to engineer a high quality DSML. The recent works in (Pérez et al., 2011; Cho et al., 2012; Sánchez-Cuadrado et al., 2012; Izquierdo et al., 2013; Kurhmann et al., 2013; Villanueva et al., 2014; Wüest et al., 2017; Barišić et al., 2018) show not only promote the validity of such strategy but also demonstrate its increasing adoption in the practice DSML engineering. Literature on engineering lifecycles is further elaborated in the next section.

The end-user collaboration from an early stage is considered as one of the fundamental agile principles in both software development (Beedle et al., 2001) and business development (Burlton et al., 2017). Implementing an agile principle is one of the requirements to be incorporated in the new agile approach proposed in this research work.

2.10 Domain-Specific Modelling Language Engineering Lifecycles

As described in the research problem (Section 1.1), the lack of agility in DSML engineering lifecycles is one of the two main problems that this research work aims to solve. This section, therefore, provides the theoretical background relevant to deepen knowledge on domain-specific modelling language engineering lifecycles.

The literature investigation revealed that one can distinguish between (a) *language-oriented* and (b) *model-oriented* engineering lifecycles. Language-oriented approaches are also known as domain-specific language (DSL) engineering life-cycles (Gabriel et al., 2011). The latter revert to the typical development process of *Software Languages Engineering* (SLE) (Kleppe, 2009). Hölldobler et al. (2017) define SLE as “the discipline of engineering software languages, which are not only applied to computer science, but to any form of domain that deals with data, their representation in form of data structures, smart systems that need control, as well as with smart services that assist us in our daily life”. Principles of DSL engineering lifecycles were first proposed to develop domain-specific programming languages (DSPLs) (Mernik et al., 2005). These principles were recently transferred to *model-oriented* approaches to engineer DSMLs (Strembeck & Zdun, 2009). Hence, the model-oriented approach refers to DSML engineering lifecycles. Nowadays, SLE investigates disciplines and systematic approaches that stretch over both *language-oriented* and *model-oriented* engineering lifecycles (Hölldobler et al., 2017).

In the following sub-sections, literature about DSL engineering lifecycles is first provided, which introduces the basic and historical principles of engineering lifecycles. Next, research findings on the most recent DSML engineering lifecycle are provided followed by their respective research advances.

2.10.1 Domain-Specific Language Engineering Lifecycles

The seminal work of Mernik et al. (2005), provides solid foundations of domain-specific language (DSL) engineering lifecycles. The authors propose five engineering phases to develop a DSL: decision, analysis (CDA – Classic Domain Analysis), design, implementation, and deployment. In addition, the work of Ceh et al. (2011) suggest to add Testing and Maintenance as final phases, resulting in a total of seven engineering phases. Figure 19 depicts the seven engineering phases in a flow chart. In the following, each phase is described.

- In the *decision phase*, decision patterns should be identified for which, in the past, developing a new DSL was profitable. According to Mernik et al. (2005), these patterns could be describing, for instance, task automation, domain-specific Analysis, Verification, Optimisation, Parallelisation, and Transformation (AVOPT).
- The *domain analysis* (CDA) phase is a detailed analysis that serves as a precondition for the design and implementation of a DSL. The objective of CDA is to determine the specific domain and collect appropriate information, which should be integrated into a coherent domain model. The latter is usually in the form of domain system properties and their dependencies (Mernik et al., 2005). Also, the systematic and organised collection of existing information should encourage the extension of information with new knowledge. CDA may be informal or more formal by following methodologies, e.g. FODA (Feature-Oriented Domain Development) (Kang et al., 1990). Notably classic domain analysis CDA can be replaced by an ontology-based domain analysis (OBDA). That is as the overall development of a DSL would benefit from an already existing domain knowledge and representation. Ceh et al. (2011) compare the two approaches and show the superiority of OBDA over CDA. Namely, OBDA is capable of what CDA is capable of doing (e.g.

providing concept vocabulary, enabling the display of property and class hierarchies, and providing a constraint mechanism), with the added advantage of reasoning and querying capabilities. These enable the validation of the ontology, and this reduces or even prevents errors in the development of the language. Ceh et al. (2011) also claim that this approach greatly helps the development of the semantics as an ontology inherently defines semantics. Moreover, already available ontologies can be employed such that the domain analysis phase can be eliminated, thus significantly reducing the time needed for language development. Ceh et al. (2011) conclude by asserting that the OBDA approach leads to minimising the cost of the DSL development as well as providing everything needed for DSL development and adds new capabilities.

- The *design* phase includes the definition of constructs and language semantics. According to Mernik et al. (2005), in this phase existing languages can be partially reused, limited or extended.
- In the *implementation* phase the language is implemented so that is ready to be used. To achieve this, Mernik et al. (2005) suggest to adopt one approach, e.g. interpreter, compiler/application generator, embedding, pre-processing, extensible compiler/interpreter, or a Commercial Off-The-Shelf (COTS) approach.
- In the *testing* phase an evaluation of the DSL is performed. According to Gabriel et al. (2011), this phase should not be skipped or relaxed as it may lead to the development of inadequate languages. The outcome of this phase can directly impact the domain analysis as adjustments or amendments might be required (see arrow back to the domain analysis phase in Figure 19).
- In the *deployment* phase the DSL and related applications or models are used.
- In the *maintenance* phase, the DSL is updated to accommodate new requirements (see loop back to the domain analysis phase in Figure 19).

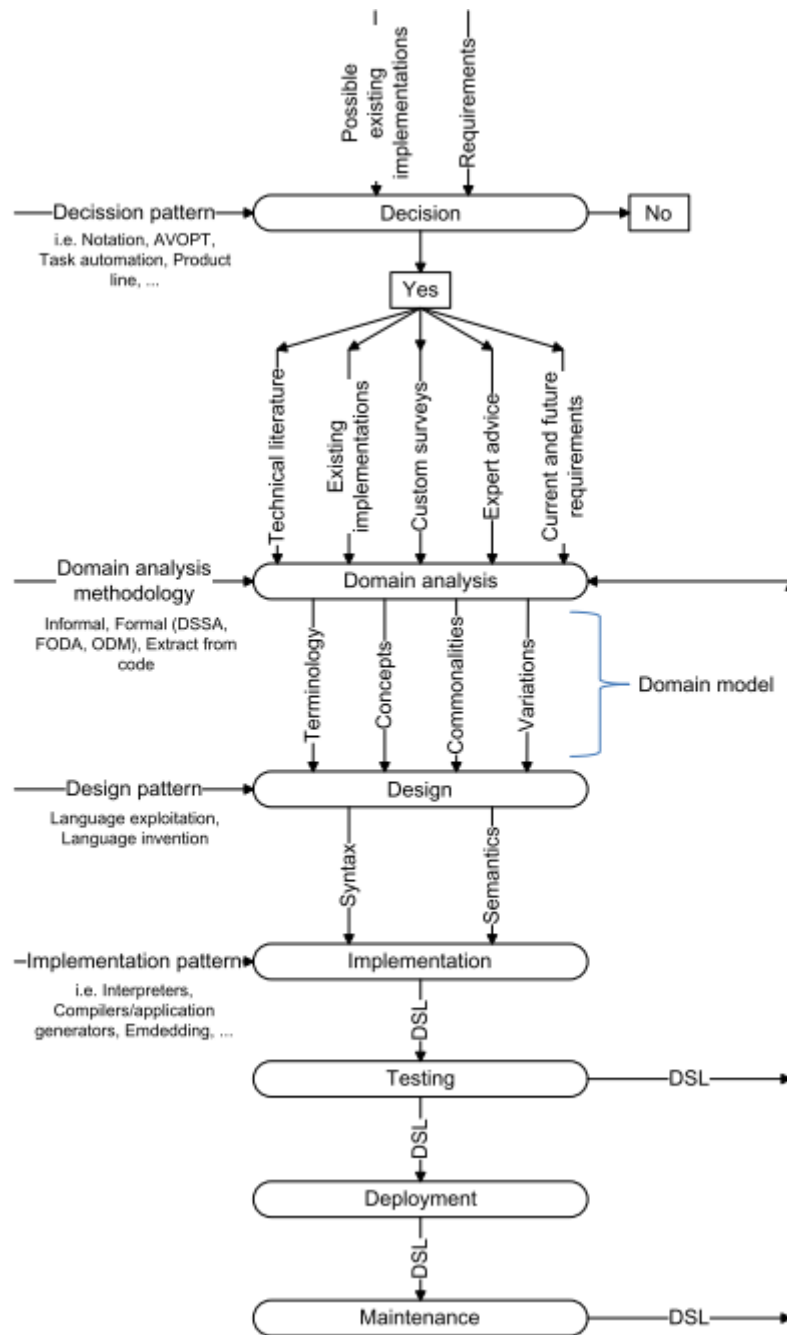


Figure 19. DSL engineering lifecycle (Ceh et al., 2011)

2.10.2 DSML Engineering Lifecycles

Differently from the DSL engineering lifecycle, a DSML engineering lifecycle includes the engineering of a meta-model, which is considered the core activity (Cho et al., 2012). A common meta-model engineering lifecycle is described in Izquierdo et al. (2013) and is depicted in Figure 20. The lifecycle spans three phases:

1. In the first phase language engineers and domain experts come together to elicit relevant domain knowledge. The cooperation between the two roles may take place in focus groups or via interviews. Domain knowledge is documented mostly in natural language.

2. The language engineers design the meta-model, i.e. abstract syntax, constraints, additional semantics.
3. Domain experts evaluate the designed meta-model. Feedback is taken into account and the meta-model may be adapted accordingly. Izquierdo et al. (2013) suggest to implement this process iteratively until the meta-model assimilates all the needed domain concepts. Only then is the correspondent DSML tool considered to be developed and ready to be used.

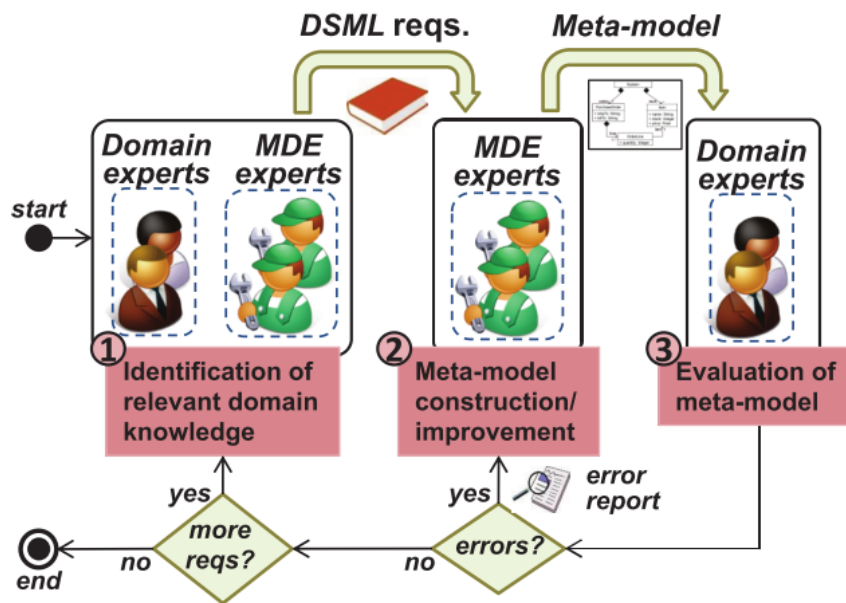


Figure 20. Meta-model Engineering Lifecycle (Izquierdo et al., 2013)

As stated in (Izquierdo et al., 2013) the lifecycle presents drawbacks in every phase.

1. In the *first phase* the domain knowledge documented in natural language is not as effective as having a knowledge in machine-readable models.
2. In the *second phase* there is lack of cooperation between end-users and language engineers as the latter just build the meta-model and then hand it over to the end-users. This is not optimal as it leads to heavy iterations until a good enough meta-model is designed.
3. In the *third phase*, the end-user is expected to evaluate the meta-model. The evaluation is rather difficult as the end-user would need an IT background to understand a class diagrams and its implications, e.g. concept of inheritance or composition (Izquierdo et al., 2013). An intuitive concrete syntax should be provided, particularly to domain experts, as well as a corresponding modelling tool. This creates the conditions for an appropriate evaluation of a DSML as the end-user can use the DSML. In fact, most of the time, amendments on the language occur only after the end-user starts designing with the DSML. Izquierdo et al. (2013) assert that, sometimes, defects on a DSML can only be found after the correspondent tool is available, in which end-users can detect missing elements. Similarly, Cho et al. (2012) claim that “an iterative and incremental DSML development process can be mundane and error-prone if there is no tool support”.

2.10.3 Advances of DSML Engineering Lifecycles

Several authors contributed to improve DSML engineering lifecycles, e.g. (Atkinson & Kuhne 2003; Stahl & Völter, 2006; Selic, 2007; Kleppe, 2009; Strembeck & Zdun, 2009; Cho et al., 2012; Izquierdo et al., 2013; Karagiannis, 2015; Barišić et al., 2018). The afore-mentioned drawbacks were addressed leading a refinement of the lifecycle (Figure 21). When comparing the latter with the lifecycle in Figure 20, at first sight one can note the increased number of engineering phases.

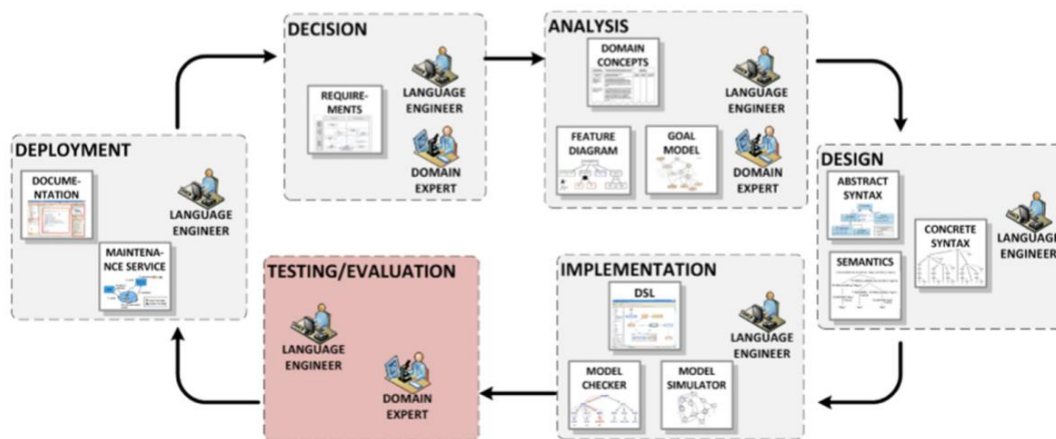


Figure 21. Typical DSML engineering life-cycle (Barišić et al., 2018)

The *decision* phase (see Figure 21) is typically the first phase of the lifecycle and has the goal to identify the need for a DSML and its validity, including justifications for its investment. The lifecycle occurs iteratively until the DSML is mature enough to be released. Multiple research works (Cho et al., 2012; Izquierdo et al., 2013; Barišić et al., 2018) propose to engage end-users in a DSML engineering lifecycle from the decision phase. This proposal aims to address the lack of cooperation between the language engineer and the domain expert (see drawback in the second phase of the lifecycle shown in the previous Section Figure 20). For instance, Barišić et al. (2018) introduced a conceptual framework supporting the engineering of DSMLs by focusing on the Usability *evaluation* (see red box in Figure 21). Barišić et al. (2018) argue that “a timely systematic approach based on User Interface experimental evaluation techniques should be used to assess the impact of DSMLs during their development process”. The purpose is to reduce the development costs from adjustments that originate from usability problems, which arise at the end of the engineering lifecycle. According to Barišić et al. (2018), in the long-run, the approach is expected to improve the productivity of the language engineers and the language users.

Analysis and *design* phases relate to the identification of relevant knowledge performed at the first phase in the previously shown lifecycle (see Figure 20). Analysis and design are sometimes considered as a single phase. In the design phase the relevant knowledge is captured in conceptual models, avoiding the issue of leaving requirements in natural language (see first drawback from phase two in previous section). Conceptual models can therefore be machine-processable. Additionally, for the design phase, Cho et al. (2012) suggest in which order the modelling language components should be developed: concrete syntax, abstract syntax and then semantics (see Figure 22).

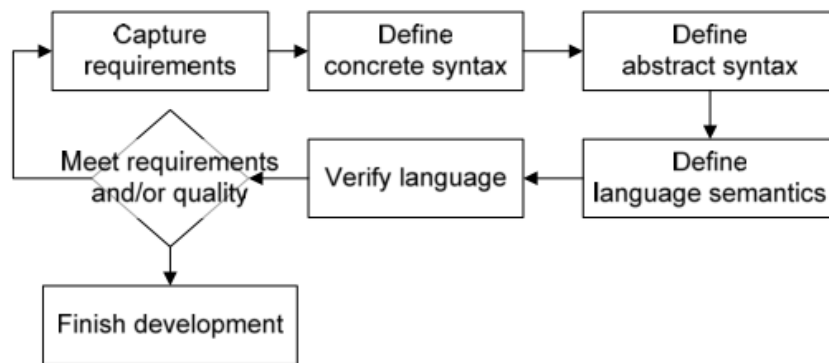


Figure 22. DSML Development Process (Cho et al., 2012)

Similarly, Strembeck and Zdun (2009) proposed several sequential and detailed sub-processes for the engineering lifecycle. The sub-processes have a different focus but are interrelated through their input and output, namely: definition of the core language, definition of the language behaviour, definition of the concrete syntax and finally the integration of the language with the platform.

As shown in Figure 21, recent lifecycles also include the *implementation* phase between the design and the evaluation phase. A language can be implemented through one of the available workbench tools such as MetaEdit+⁴ (Kelly, Lyytinen, & Rossi, 2013), ADOxx⁵ (Fill & Karagiannis, 2013), Cubetto⁶ or Eclipse Modelling Framework (EMF) (Moore, 2004), DiaGen⁷, the Generic Modelling Environment (Ledeczi et al., 2001), and Eclipse-based tools like ATL⁸ (Jouault, Allilaire, Bézivin, & Kurtev, 2008), Eugenia⁹, GMF¹⁰ (Gronback, 2009) and Sirius¹¹, and AToMPM¹² (Syriani et al., 2013). The outcome of this phase addresses the drawback saw above in phase (3). In fact, during the implementation phase, constructs from the abstract syntax are furnished with graphical notations, which facilitate domain experts in the testing phase (see testing phase in Figure 21). Workbench tools typically also provide a modelling environment in which models can be designed, thus the language can be evaluated before the deployment phase.

In the *deployment* phase, a new modelling tool, tailored to the DSML is created (see deployment phase in Figure 21). If already existing, the tool is updated to incorporate the new version of the DSML.

Frank (2013a) proposes a macro process with seven steps for a DSML engineering (see lifecycle in Figure 23): (1) clarification of scope and purpose, (2) analysis of general requirements, (3) derivation of specific requirements using a set of scenarios, (4) specification of language (abstract syntax), (5) provision of a graphical notation (concrete syntax), (6) optional development of a modelling tool, and (7) evaluation and iterative refinement of developed artefacts.

⁴ <http://www.metacase.com/products.html>

⁵ <https://www.adoxx.org/live/home>

⁶ <http://www.semture.de>

⁷ <http://www2.cs.unibw.de/tools/DiaGen/>

⁸ <https://www.eclipse.org/atl/>

⁹ <https://www.eclipse.org/epsilon/doc/eugenia/>

¹⁰ <https://www.eclipse.org/modelling/gmp/>

¹¹ <https://www.eclipse.org/sirius/>

¹² <https://atopmpm.github.io/>

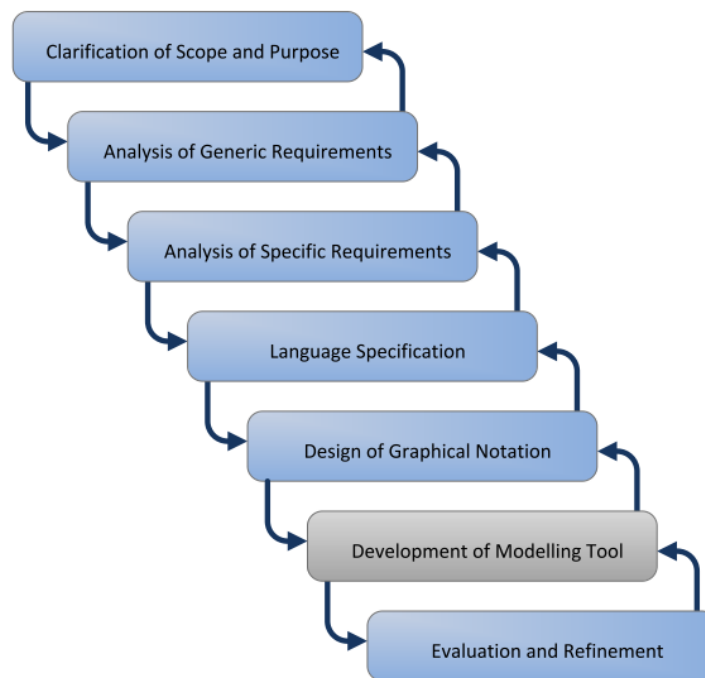


Figure 23 The seven steps in the macro process (Frank, 2013a)

Different engineering lifecycles differ in some phases. For instance, the decision phase may be considered in the design phase in some approaches or not considered at all. The lifecycle of Gabriel et al. (2011) starts with the domain analysis, Barišić et al. (2018) starts a step earlier with “decision”, and Frank (2013a) another step earlier with “clarification of scope and purpose”. In other approaches both the domain analysis and design phase may be considered in the same phase albeit with different objectives. For instance, in the work of Kleppe (2009) the domain analysis aims to elicit domain concepts whilst the design phase is devoted to capture concepts and relationship and shape the conceptual model.

Further research investigation about DSML engineering lifecycles revealed a common agreement on considering *design* (or conceptualisation), *implementation* and *evaluation* as the three essential phases when engineering DSMLs (Gabriel et al., 2011; Chiprianov et al., 2013). These phases share the characteristic of being sequential to one another. Therefore, the implementation cannot start before the conceptualisation has been completed. Consequently, the evaluation phase that cannot start before the DSML has been completely implemented. This demonstrates a lack of agility in the existing DSML engineering lifecycles.

In response to such lack of agility, Karagiannis (2015) proposed the AMME Lifecycle, which is described in the following section.

2.10.4 Agile Principles for DSML Engineering Lifecycles and Considerations

Agile principles have been recently applied in DSML engineering lifecycle by Karagiannis, (2015). The AMME Lifecycle (Karagiannis, 2015; Karagiannis, 2018; Bork et al., 2019) is also

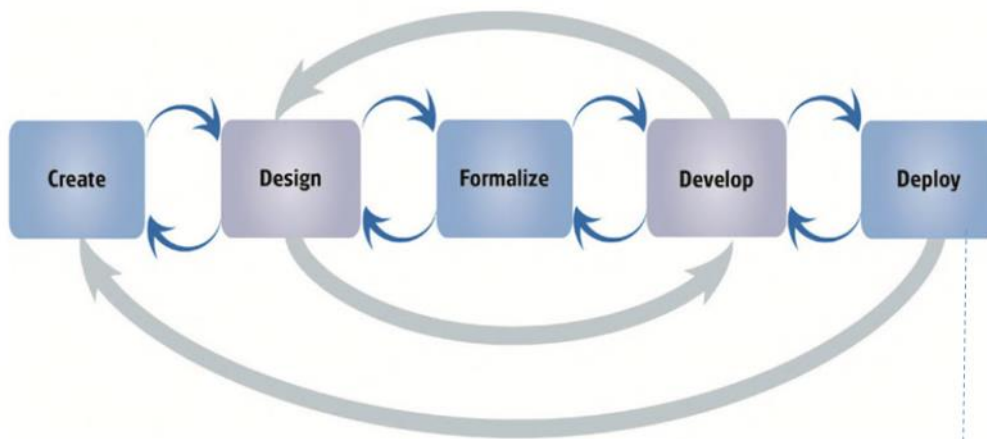
called the OMiLab lifecycle in (Karagiannis, 2015; Karagiannis, 2018) and has been introduced as an agile management approach in response to the rigid DSML engineering lifecycles. Like in software engineering, where the principles in the Agile Manifesto (Beedle et al., 2001) aimed to overcome the shortcomings of sequential approaches (e.g. waterfall), the AMME Lifecycle aims to keep up with changes in modelling requirements and deliver DSMLs in a timely manner. Specifically, the principles from the Agile Manifesto are applied to the fundamentals of the modelling method framework (Karagiannis & Kühn, 2002) (see framework in Section 2.2). The applied agile principles described in (Karagiannis, 2015) are the following:

- *Iterative development* - work cycles allow for revisiting the same work items;
- *Incremental development* - successive usable versions are built upon previous versions;
- *Version control* - the output of each iteration is traceable across multiple versions and in relation to their requirements;
- *Team control* - small groups of people are assigned to backlog items with shared accountability.

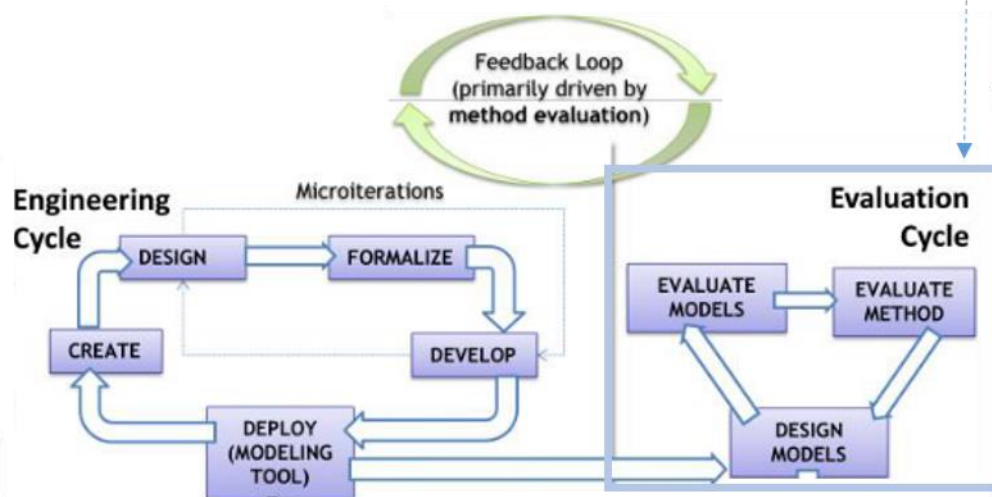
The AMME Lifecycle introduces feedback channels along five phases (see backward arrows at the bottom of Figure 24) with the aim of supporting the engineering process during the propagation and evolution of modelling requirements. Consequently, various DSMLs were created following this agile modelling method engineering (Karagiannis et al., 2016; Buchmann, 2016).

Karagiannis (2015) describes the five phases as follows (see bottom of Figure 24):

1. The *Creation phase* is a mix of knowledge acquisition and requirements elicitation activities that capture and represent the modelling requirements;
2. The *Design phase* specifies the meta-model, language grammar, notation and functionality;
3. The *Formalise phase* aims to describe the outcome of the previous phase in non-ambiguous representations with the purpose of sharing results within a scientific community;
4. The *Develop phase* produces concrete modelling prototypes;
5. The *Deploy/Validate phase* deals with the deployment of the modelling language in a modelling tool as well as the use of the language in the tool for validation purposes. Bork et al., (2019) further elaborates on the Deploy/Validate phase by stressing the distinction between the engineering and modelling component (see engineering cycle and evaluation cycle in Figure 24, respectively). Hence, a modelling language has to be deployed in a modelling tool before it can be used for modelling, and thus for evaluation.



(Karagiannis, 2018; Efendioglu et al., 2017; Karagiannis 2015)



(Bork et al., 2019)

Figure 24. The AMME (or OMiLab) Lifecycle. Taken from (Bork et al., 2019; Karagiannis, 2018; Efendioglu et al., 2017; Karagiannis 2015)

As already reported in the research problem (Sub-section 1.1.2), the AMME Lifecycle distinguishes between two components: the *language engineering component* (see engineering cycle in Figure 24) and the *modelling component* (see evaluation cycle in Figure 24). The two components are typically implemented in a meta-modelling and a modelling tool, respectively. For instance, this is the case of the already mentioned ADOxx and MetaEdit, which are among widely adopted in research.

On the one hand, this approach promotes the correct creation of a DSML, and on the other hand it prevents a DSML from being used for evaluation purposes before the deployment has been accomplished. Additionally, the engineering component does not involve the domain expert (or as above-mentioned stakeholders) until the DSML is deployed. As described already (in Sub-section 1.1.2 and Section 2.9), this aspect is in contradiction of the agile principle of involving an end-user since an early stage of a lifecycle.

Another consideration concerns the engineering phases of the AMME Lifecycle. Although the phases can intertwine (i.e. create-design, design-formalize and design-develop), if a new

requirement is to be integrated it has to be propagated through all the phases, sequentially. (Efendioglu et al., 2017). First, the new requirement needs to be captured and represented in the form of modelling requirement in the creation phase. Next, the requirement is designed such that it fits other modelling constructs. The formalisation phase may be skipped. Next, the new version of the DSML is implemented in a meta-modelling tool (development phase). Then, the DSML can be deployed to release a modelling tool tailored to the new version of the DSML. Finally, the new DSML can be used to design models. In this case also, the correct creation of a DSML is promoted. However, going through all the engineering phases subsequently is time-consuming. Moreover, the longer the propagation of a new requirement the higher the risk to have outdated requirements (Hepp et al., 2005).

Currently, the AMME Lifecycle is the most advanced lifecycle within Enterprise Modelling striving to incorporate the agile principles that revert to the Agile Manifesto (Beedle et al., 2001). As discussed, however, there is still room to inject more agility into DSML engineering lifecycle. It is the objective of this research work to propose an agile approach that avoids sequential engineering phases and promotes the cooperation between the language engineer and the domain expert.

In the next section, the concept of agility is further elaborated on the context of enterprises.

2.11 Agility in Enterprises: Definitions, Principles, Trends and New Challenges

Giachetti (2010) defines an enterprise as a “complex socio-technical system that comprises interdependent resources of people, information, and technology that must interact with each other and their environment in support of a common mission”. Enterprises are increasingly characterised by high competition, cross-organisation cooperation, and continuous and unexpected change. According to Hinkelmann et al., (2016) “ability of keeping up with continuous and unexpected change is an essential quality of modern enterprises and will become a necessity for existence”. Kidd (1994) coins this ability as “agility” and defines it as the rapid and proactive adaptation of enterprise elements to unexpected and unpredicted changes. Similarly, Dove (1999) defines agility as “the ability of an organisation to thrive in a continuously changing, unpredictable business environment”. For Bohdana et al. (2007), an agile enterprise is synonym of “adaptable” and “flexible” enterprise. “Adaptability” and “flexibility”, together with “sensitivity”, “responsiveness”, “autonomy”, and “interoperability” are defined by Horkoff et al., (2016) as emerging quality attributes for advanced enterprise models. Horkoff et al., (2016) assert that addressing these quality attributes “will allow all components of an enterprise to operate together in the cooperative manner for the purpose of maximising overall benefit to the enterprise”.

The pioneers of agile principles are Beedle et al., (2001), who proposed the Agile Manifesto in response to the inability of software engineering to deliver in a timely manner. Several agile methodologies have been thereafter conceived, which embrace these agile principles. Among the widely used methodologies there are Scrum (Schwaber & Beedle, 2002), eXtreme Programming (XP) (Beck & Kent, 2000) and the Lean Software Development (Poppendieck, 2007).

In their recent work, Burlton, Ross and Zachmann (2017) argue that it is no longer sufficient to apply agile principles on software development or organisational schemes only. According to Burlton et al., (2017) agile principles should instead be extended to the whole business. Thus, they refer to “Business Agility” as the “ability to modify dynamically the concepts and structures of the business itself for maintaining relevance in the context of a dramatically changing, complex and uncertain operational environment” (Burlton et al., 2017). In support of this view, Burlton et al., (2017) formulated ten principles in what they call “*The Business Agility Manifesto*”. The principles are summarised below and supplied with further literature.

1. **Perpetual Changes.** This principle refers to the need within a business for accommodating unceasing innovation and fast-paced change. Change should be scalable and sustainable.
2. **Business Solution Agility.** This principle stresses the insufficiency of faster software development in keeping up with continuous and rapid change. Burlton et al., (2017) argue that we are in the “Knowledge Age”, where models are used to retain business knowledge. Nonaka et al. (2000) state that the capability to create and use knowledge is considered as the source of sustainable competitive advantages. Drucker (2001) predicted that knowledge will replace machinery, equipment, capital, raw material and labour to become the most important factor. Burlton et al. (2017) sustain that an agile business solution is measured on how much knowledge is configured into it and how easily that knowledge can be changed or reconfigured. This action can be enabled by models having the above-mentioned quality attributes (Horkoff et al., 2016). Burlton et al., (2017) further argue that business knowledge is to be deployed into business processes and products in a timely, effective, selective, repeatable, pervasive, traceable, and retractable manner.

3. **Business Value Creation.** In this principle business value must be the over-riding justification for expending resources for all investments. Knowledge should be such that scrap and rework can be performed avoiding unintended consequences. Burlton et al., (2017) suggest to use the value chain model for planning and managing value creation. This can be then used to design, change, manage, operate, and analyse the value chain.
4. **Value Chain.** This principle stresses the importance of stakeholders in the value chain as well as the importance of concept models representing end-to-end processes in a value chain. Knowledge about dependencies in a value chain enables the prediction of the impacts of change as well as contributing to the scoping of the boundaries of business design and software implementation initiatives.
5. **Business Knowledge.** This principle refers to the importance of a business knowledge and distinguish it from the limited software development knowledge. Also, business knowledge should have a shared understanding in order to be effectively and unambiguously communicated. Concept models should be used to retain this knowledge and to enable its reuse.
6. **Business Knowledge Management.** This principle focuses on the importance of expressing business knowledge explicitly in a form that is accessible to all business audience. Namely, knowledge should be explicit, accessible, protected, sharable, reusable, retainable, and updatable. This makes for the effective management of business knowledge.
7. **Business Knowledge-Base.** This principle defines the business knowledge-base as the end result of transforming tacit (mental) knowledge to explicit (stored, common, sharable) knowledge. The knowledge base should support design decisions in configuring business processes and products. The latter is essential for both automation and for assembling business capabilities. The knowledge-base is considered a business asset and not a mere implementation of technology.
8. **Single Source of Business Truth.** This principle refers to the importance of the Business Knowledge-Base for the business concept definitions and business policy decisions. Additionally, the separation of concerns in the Business Knowledge-Base is a must, i.e. business capabilities or business solutions. For example, separating concepts from business rules, business rules from processes, processes from events, and roles from all technical design descriptions.
9. **Business Integrity.** This principle stresses the importance of consistency in business results towards external stakeholders. Conceptual models properly design the structure of business concepts and thus ensure consistency in business results. The design of conceptual models avoids needless subsequent redesign and reconstruction of databases and systems. Consistency should also embrace behaviour and repeatable operational decisions, business rules. Else, data quality problems can arise.
10. **Business Strategy.** This principle focuses on the importance of managing business knowledge. The latter provides a huge competitive advantage if used to execute the business strategy and to configure business processes and products. Hence, the need for explicit business knowledge is emphasised in this principle as well. Business knowledge is needed by every new business or technology channel.

As we can see, knowledge plays a fundamental role in solutions that want to embrace business agility principles. In their visionary paper, Sandkuhl et al. (2018) predict that such knowledge will be captured via enterprise modelling by the majority of stakeholders:

“Ten years from now, the majority of organisational stakeholders uses enterprise modelling (often without noticing it) to capture, store, distribute, integrate and retrieve essential knowledge relevant for their local practices in a way that supports long-term, cross- concern organisational objectives”.

According to Horkoff et al. (2018) the remarkable effort required in the creation, retention and use of knowledge is currently cumbersome to business agility. Approaches, modelling methods and tools tend remain formal and inflexible at present (Bork & Alter, 2018).

Sandkuhl et al. (2018) claim the need for lightweight approaches that shift from the traditional Enterprise Modelling qualities such as completeness and coherence to usefulness and impact. In doing so, such approaches should be able not only to support specialised roles like architects or IT specialists, but can open up to a wider range of organisational stakeholders who can benefit from models for analysis, design and, or decision problems (Sandkuhl et al., 2018). In line with this argument, Sandkuhl et al., (2018), predict that modelling will be incorporated in every day work wherein “non-experts in modelling will do modelling, even without knowing it”. This hypothesis gains confidence from the work of Braun et al., (2015), who showed that the practice of modelling by domain experts is a reality in some application areas of healthcare, i.e. in some hospital physicians and quality managers who are involved in process modelling.

This trend, however, comes with the expenses of a high synergy between domain experts and modelling experts, which might be time consuming and quite a demanding engineering effort, and thus a further threat to agility. Sandkuhl et al., (2018) propose to tackle this problem with assistive technologies, which facilitate the practice of modelling. As an example, in Laurenzi et al., (2019) an assistive and pattern learning-driven process modelling approach is proposed. The aim of Laurenzi et al., (2019) is to create good quality BPMN process models while relieving the burden of the intensive cooperation between modelling and domain experts. To achieve this, machine-interpretable knowledge is employed. Similarly, machine-interpretable knowledge is employed by Delfmann et al., (2010) to investigate model patterns, and by Fill (2011) to interpret the meaning of labels and detect similar constructs in other models. Machine-interpretable knowledge (Hinkelmann, Gerber, et al., 2016) is concerned with the type of knowledge representation that is interpretable by machines and reverts the Artificial Intelligence sub-field Knowledge Representation and Reasoning (Van Harmelen, Lifschitz, & Porter, 2008). Machine-interpretable knowledge allows automation and enables machines to solve complex problems. In order for assistive technologies like recommendation systems to be intelligent, they have to build upon a knowledge-base that is machine-interpretable. The intelligence lays on the ability of machines to explain how certain recommendations were deduced (Van Harmelen & Ten Teije, 2019). Hinkelmann, Gerber, et al., (2016) regard the continuous alignment between graphical enterprise models and the underlying machine-interpretable knowledge as one of the innovative grand challenges of today’s enterprise. Mastering this challenge means that graphical models, which are interpretable by people, are continuously synchronised with an underlying knowledge, which is interpretable by machines. Such continuous alignment is thereby key aspect in fostering agility and enabling the creation of the intelligent enterprises of the future.

The new agile approach should be able to seamlessly align the graphical and machine-interpretable representation of a modelling language.

The objective stimulated further investigation of the literature on approaches that foresee the adoption of machine-interpretable knowledge within the context of Enterprise Modelling. The investigation included approaches for the alignment of graphical and machine-

interpretable representation of modelling languages. An overview of these research findings is reported in the next section.

2.12 Machine-Interpretable Knowledge and Enterprise Modelling

Graphical notations facilitate the human understanding and interpretability of models. When models are used for automations and operations purposes, the model should be machine-interpretable or at least machine-readable. A machine-interpretable representation of knowledge enables decision making, analysis, adaptation and evaluation (Hinkelmann *et al.*, 2018). In business process automation, for instance, process models determine the workflow executed by the workflow engine. For decision-making purposes, it is common practice to work with models, such as the Decision Model and Notation (OMG 2016). According to Hinkelmann, Gerber, et al. (2016), there is a distinction between machine- interpretable models and machine-readable models. The former are represented in a format on which reasoning can be performed. Hence, machine-interpretable models can turn passive data storage into an active device.

Logic-based languages such as ontologies (Guizzardi, 2007; Guizzardi, 2012) can be used to express knowledge in a machine-interpretable format. The following sub-sections introduce first the definition of an ontology. Then, the two most common and widely-used ontology languages are described. Next, recent work that successfully used ontologies in the context of Enterprise Modelling is provided. Finally, research advances on approaches that foresee the combination of ontologies with models and modelling languages are described.

2.12.1 Ontology Definition

Etymologically, the term “ontology” means the study of existence and comes from philosophy (Guizzardi, 2007). The term reverts to the attempt of Aristotle of classifying the things in the world. In computer and information science, an ontology assists in specifying and clarifying the concepts employed in specific domains and helps formalising them within the framework of a formal theory with a well-understood logical (syntactic and semantic) structure (Guizzardi, 2012). In Artificial Intelligence (AI) the term “ontology” is used to describe what can be (computationally) represented of the world in a program (Studer, Benjamins, & Fensel, 1998) or in a model. This representation enables reasoning about models of the world.

Ontology is a fundamental concept in knowledge representation (Van Harmelen et al., 2008). Several definitions about ontologies exist in the literature (Gruber, 1993; Guarino et al., 1995; Russ 1997; Guarino, 1998; Studer et al., 1998; Staab & Studer 2009; Guizzardi 2007; Gomez-Perez et al., 2004). Definitions have changed and evolved over the years. One of the very first and widely cited definition was given by Gruber (1993), who stated that an “ontology is a specification of a conceptualisation”. Based on this definition Studer et al. (1998) defined an ontology as a “[...] formal, explicit specification of a shared conceptualisation”.

In Gašević et al. (2009), the authors further elaborate the two keywords “specification” and “conceptualisation” of this definition. Conceptualisation represents an abstract and simplified view of the world. Thus, it addresses a particular domain. Every conceptualisation is based on: concepts, objects, entities assumed to exist in an area of interest and relationships among them. The term specification, on the other hand, means a formal and declarative representation of the conceptualisation. The type of concept and the constraints in the ontology’s data structure are stated in a declarative and explicit way using a formal language. Such formal representation allows an ontology to be machine-interpretable and although it cannot be run as a program, it represents some declarative knowledge to be used by programs.

Ontologies are expressed in ontology languages. Well-known and widely adopted ontology languages relate to the concept of Semantic Web: “Semantic web originated from the vision that machines are enabled to conduct automated reasoning and can thus infer information from resources on the world-wide-web” (Berners-Lee, Hendler, & Lassila, 2001). Within the vision of Semantic Web classes, properties and instances are all web-based resources. Two prominent ontology languages exist (Allemang & Hendler, 2011): the Resource Description Framework Schema¹³ (RDF(S)) and the Ontology Web Language¹⁴ (OWL). The following sub-section describes them in detail.

2.12.2 Prominent Ontology Languages: RDF(S) and OWL

The ontology language RDF(S) (W3C, 2014c) builds upon the Resource Description Framework (RDF) (W3C, 2004b) and is in compliance with its formal semantics specified in W3C (2014b). RDF is a data model that provides a way to express simple statements about resources by using properties and values. RDF(S) is a (W3C) recommendation and the latest specification version is RDF(S) 1.1 (W3C, 2014c). The abstract syntax of this specification provides structure to web (or RDF) resources so that they can be expressed in the form of triples: *subject-predicate-object*. A set of triples forms an RDF graph, which is typically stored in a knowledge graph database (also known as triplestore). RDF(S) is used to define object-oriented concepts such as *classes* and *properties*.

Specifically, in RDF(S) any resource can be specified as either a class, or a property or both. RDF(S) extends RDF by providing a set of classes and properties that allow for a more expressive description of RDF resources. For instance, the property *rdfs:subClassOf* is a specialisation relationship between classes, which allows the definition of a taxonomy of classes; *rdfs:domain* and *rdfs:range* allow the specification of the source and target of a property, i.e. a relation between two classes or a relation between one class and one datatype like *xsd:string*, *xsd:Boolean*, *xsd:integer*. Datatypes denote the built-in datatypes defined in XML Schema Definition Language (XSD) (W3C, 2012) and a restricted list of it is recommended by the W3C as RDF-compatible (see (W3C, 2014a). When a resource is a member of a class, the resource is an *instance* of the specified class. The membership of a class is indicated with the property name *rdf:type*. At hand, RDF(S) defines RDF vocabularies and for this are considered to be supportive in the ontology specification. Additionally, the W3C recommendation (W3C, 2014b) defines 13 RDF(S) entailments patterns, which interprets both RDF and RDF(S) vocabulary that correspond to inference rules. One example of such entailment patterns is that if a class “uuu” is sub-class of the class “xxx”, and the resource “vvv” is a member of the class “uuu”, it is then deduced that “vvv” is also a member of the class “xxx”. This entailment pattern corresponds to rule “rdfs9” in W3C (2014b). Therefore, at hand, in addition to RDF, RDF(S) can define some taxonomies and can do some basic inference about them.

OWL (W3C OWL Working Group, 2009) builds on RDF(s). It adds more expressivity power to RDF(S) by injecting a set of axioms that further constraints the ontology language RDF(S). These axioms allow the equivalence or disjointness between classes, or the numerical limitation (i.e. cardinality) of a particular property to be stated. Three decreasingly expressive sublanguages of OWL exist: OWL-Full, OWL-DL and OWL-Lite (Baader, 2011). OWL-Full is fully compatible with RDF(S) as all the primitives available in RDF and RDF(S) are available and usable in OWL-Full. This makes the latter the most expressive version of OWL but also with the main drawback that the computational decidability (i.e. all computations will

¹³ <https://www.w3.org/TR/rdf-schema/>

¹⁴ <https://www.w3.org/OWL/>

finish in a finite time) cannot be retained (Horrocks, Patel-Schneider, & van Harmelen, 2003). Thus, the reasoning support is rather unpredictable. This reverts to the basic undecidable reasoning problems in the First-Order Logic (FOL) (Staab & Studer, 2009). To overcome the decidability issue, OWL-DL was developed as a sub-language of OWL-Full. The OWL-DL has a direct correspondence to the Description Logics (Baader & Nutt 2003), which is considered as a decidable subset of FOPL. The knowledge representation in OWL-DL is restricted to two levels: the TBox (i.e. containing the declaration of classes) and the ABox (i.e. containing individuals of classes defined in the TBox). The restricted representation comes with limitations to ensure decidability. For instance, a class cannot be an instance at the same time. OWL-Lite is a further restriction of OWL-DL, which aims to lower the language complexity of the latter. There exist further sub-languages of OWL-DL featuring more limitations. A comprehensive description of the three OWL sub-languages can be found in (McGuinness & van Harmelen, 2014).

Different kinds of reasoning can be applied, depending on the expressivity of the ontology language. Ontologies expressed in RDF(S), for example, can be combined with semantic rules to draw new insights from the ontology. Conversely, given the higher expressivity of OWL, reasoning services are directly applied over the constrained ontology, e.g. for classifying individuals or instances.

2.12.3 Ontologies in Enterprise Modelling

The term ontology is often used to refer to semantic technologies (Allemang & Hendler, 2011), which includes not only the representation of facts (or as above-mentioned web-based resources) but also semantic rules and inference engines for reasoning services.

Semantic technologies are widely adopted in enterprise modelling and make use of ontologies for describing models in an ontology format (Hinkelmann et al., 2013; Emmenegger et al., 2013), for transforming model into ontologies (Rospocher et al. 2014; Emmenegger et al. 2013; Natschläger 2011; Thomas & Fellmann 2007), or to semantically annotate models (Fill, Schremser, Karagiannis, 2013).

The purpose for the use of semantic technologies can vary. Recent works that successfully implemented semantic technologies are subsequently reported.

Gailly et al., (2017) present a recommendation-based conceptual modelling and ontology evolution framework (CMOE+) that makes use of enterprise-specific ontologies. The latter are used as reference to overcome the lack of consistency and interoperability in the creation of models.

Martin, (2016) developed a new ontology-based case-based reasoning (OBCBR) approach, to facilitate accessibility of experiential knowledge obtained from previous cases or projects.

Emmenegger et al., (2017) transferred the OBCBR approach conceived in Martin (2016) to the workplace learning application area. The approach aims to support inexperienced employees in public administrations by suggesting historical cases and providing recommendations of experts and learning resources.

Walter et al., (2014) use ontologies with reasoning services to support the domain-specific modelling language designers and users through the development and usage of DSMLs. Designers benefit from constraint analysis whereas users benefit from progressive verification, debugging, support and assisted modelling. for addressing constrain definition, progressive evaluation, suggestions and debugging.

Research by Rospocher et al. (2014) presents a formal ontological description of the language standard BPMN (i.e. the BPMN Ontology). Reasoning services such as consistency checking and query answering are applied to detect the compliance of process models with the BPMN specification.

Emmenegger et al. (2013) developed an Early-Warning-System (EWS) that integrates semantic technologies for the assessment of procurement risks. Enterprise ontology that reflects the standard ArchiMate (i.e. ArchiMEO (Hinkelmann et al., 2013)) is extended to represent procurements risks. Semantic rules are used together with the extended ontology to assess and monitor procurement risks in real-time.

Thönssen and Lutz (2013) used semantic technologies to automate and improve a contract's lifecycle management. Information like contract begin, contract end, contract parties and obligations were automatically extracted using automated metadata generation.

2.12.4 Approaches combining Ontologies with Modelling Languages

Approaches that strive to combine graphical modelling languages with ontologies are characterised by two kinds of semantics: one that is defined in the meta-model and one that is defined in the ontology.

Abmann, Zschaler and Wagner (2006) assume that model-based approaches and ontologies were developed in isolation and investigate the role of ontologies, models, and meta-models to bridge the gap between the model-driven engineering (MDE) and Semantic Web, respectively.

Höfferer (2007) discusses the relationship between meta-models and ontologies stating that “[...] there is no a commonly agreed view on the relationship between meta-models and ontologies in the scientific community”. Moreover, he emphasises that meta-models and ontologies are different but complementary concepts.

Sutii et al., (2014) offer a systematic literature review, and report similarities and differences between ontologies and domain-specific languages, including meta-models. Similarly to Höfferer (2007), Sutii et al. (2014) conclude that both ontologies and meta-models have complementary benefits, which make them suitable for combination.

It is common to combine concepts of meta-models and their instances with formal semantics, i.e. ontology concepts and instances, respectively (Kappel et al., 2006; Dietz, 2006; Fill & Burzynski, 2009). The main purpose of the combination is to supply meta-models and models with formal semantics. Namely, ontologies provide (1) the semantics of the modelling language constructs, and (2) the semantics of model instances (Kramler et al., 2006; Hrgovic et al., 2013).

The practice of combining meta-models and models with ontologies is known as semantic lifting (Hinkelmann, Gerber, et al., 2016). Semantic lifting is defined as “[...] the process of associating content items with suitable semantic objects as metadata to turn ‘unstructured’ content items into semantic knowledge resources” (Azzini et al., 2013). Semantic lifting is also known as semantic annotation (Liao et al., 2015; Fill et al., 2013). In these approaches, the ontologies are independent from the concepts of the human-interpretable, graphical languages. The ontology comprises class definitions which represent the formal semantics of modelling elements. The class definitions serve to annotate modelling elements (i.e. in the meta-model layer) and model elements (i.e. in the model layer). The basis for interoperability is provided by linking elements of the models and meta-models with ontology concepts.

Figure 25 shows the conceptual architecture for semantic lifting. Different model types in the enterprise architecture are created which correspond to different meta-models. The meta-models primarily define syntactical aspects as well as certain semantic aspects of model elements. The ontologies define the machine-interpretable semantics of the modelling concepts. This approach requires the relationship between the graphical and the machine-interpretable modelling language to be defined (Hrgovic et al., 2013).

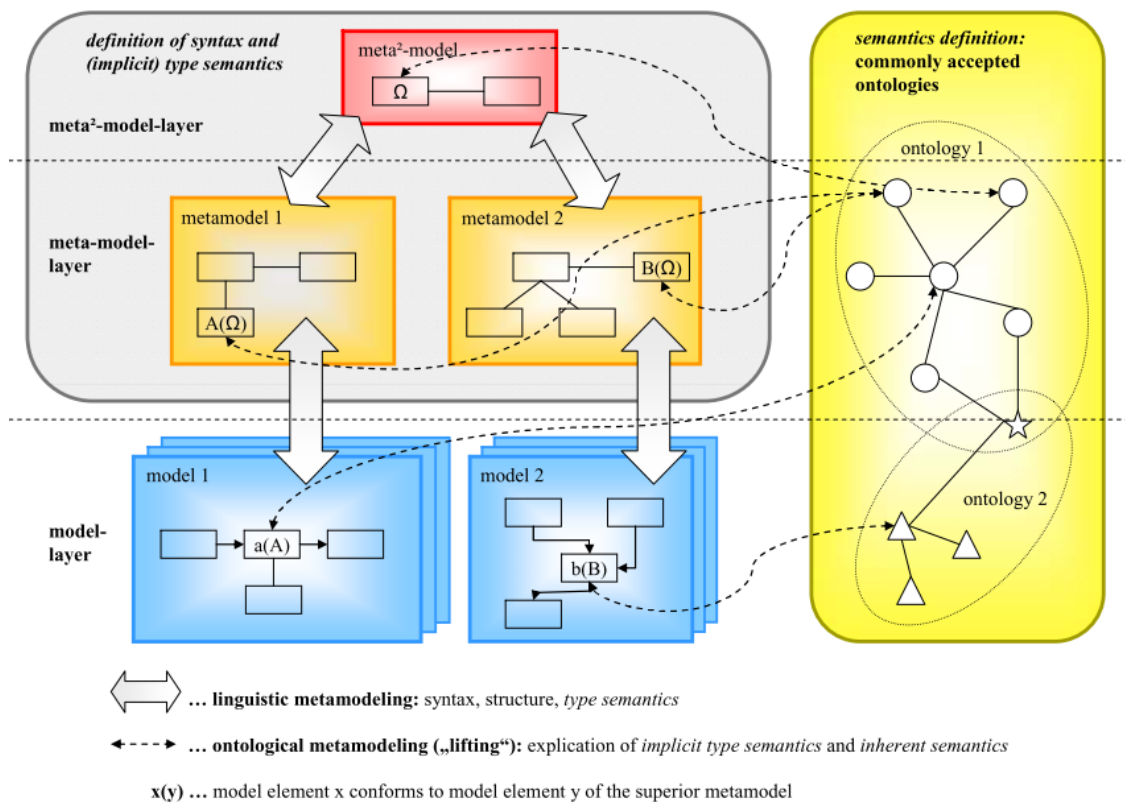


Figure 25. Architecture for semantic interoperability using both metamodels and ontologies

Just as with the use of ontologies, the application domains for the semantic lifting can vary. For instance, business-IT alignment (Nicola et al., 2008), process mining (Azzini et al., 2013), learning (De Angelis et al., 2016) and cloud service specifications (Hinkelmann, Laurenzi et al., 2016).

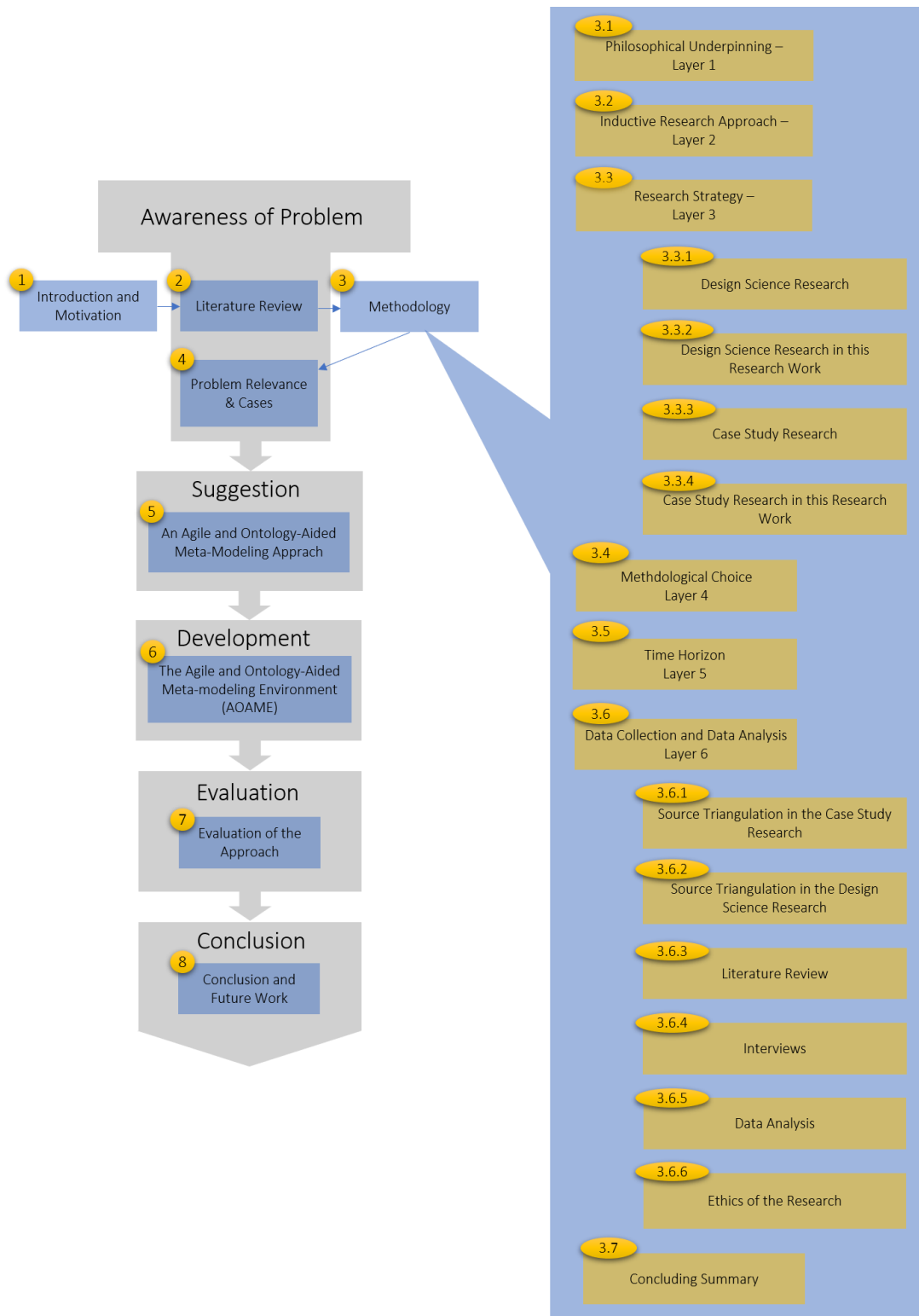
The drawback of this approach, however, lies in the consistency of the semantics between (meta) models and their representation in ontologies. Keeping them separate tends to cause incompatible semantics. To avoid that, the manual or semi-automatic alignment between meta-model concepts and ontology concepts has to be performed. However, this might be error-prone and time-consuming. Such drawbacks could be alleviated by mechanisms for the automatic generation of ontologies from models. Examples of such mechanisms range from the creation of knowledge graphs (Karagiannis & Buchmann, 2018), to more expressive ontologies, e.g. OWL-FA (Emmenegger et al., 2017). Nevertheless, the consistency issue remains due to the strictly separate representations. For instance, if a change occurs in the ontology, the human-interpretable model has to be adapted accordingly. Also, if changes occur in the meta-model, transformation patterns for the ontology generation might need to be adapted.

It is objective of this research to provide an agile approach able to preserve the seamless consistency between the graphical and machine-interpretable representation of a modelling language after changes occur on one another.

2.13 Concluding Summary

This chapter provides the theoretical background for the relevant aspects introduced in the research problem. The chapter starts by framing the context of the research within the Enterprise Modelling research discipline. Definitions for models and modelling languages are provided together with the widely adopted technique meta-modelling to create modelling languages. Some background about the notion model-based approach is also provided before diving into the definition of a DSML. For this, a comparison between DSMLs and GPMLs is presented, including a practical example. Then, terminology definitions about DSL, DSML and DSPL are given, in which the notion “domain” is clarified. Next, several application areas for DSMLs are reported which focus either on software development or knowledge retention. After that, the different roles around DSMLs are introduced as it clarifies the importance of the language engineers and domain experts in approaches for DSML engineering. Among these approaches, it was identified that domain-specific adaptations of modelling language have greater benefits than dedicated DSMLs. Therefore, the literature investigation explored the common techniques for domain-specific adaptations by analysing strengths and weaknesses of each of the techniques. It resulted that the meta-model customisation is the most appropriate technique as it supports agility in DSML engineering. Therefore, it is in line with the research objective of conceiving an agile approach. The main challenges that DSML engineering face are then reported together with related work that strive to overcome them. Next, an overview about existing domain-specific modelling language engineering lifecycles and their advances is provided. The current *lack of agility* was then elaborated by analysing the AMME Lifecycle, which is the only example striving to embrace agile principles. The first problem this research work aims to solve was thereby framed. Subsequently, to increase understanding around the concept of agility in enterprises, an overview is provided containing definitions, agile principles and trends, as well as new challenges. Findings led to the conclusion that knowledge is key enabler of agile approaches. Moreover, the importance of having a machine-interpretable knowledge was emphasised as it provides the basis to build intelligent information systems and, in turn, intelligent enterprises. Within this context, one of the major challenges is the continuous alignment between graphical and machine-interpretable representations of knowledge. Hence, this investigation aimed to deepen understanding on approaches that allow for such alignment. Findings show that current approaches separate the modelling languages and models from ontologies (i.e. machine-interpretable knowledge), which causes *inconsistencies* when changes occur on one another. The second problem that this research work aims to solve was thereby framed.

3. METHODOLOGY



This chapter describes the scientific procedure that is embraced to address this research work. The description of the chapter is underpinned by the “research onion” proposed by Saunders, Lewis, and Thornihill (2009). Figure 26 depicts the research onion and it contains six layers (starting from the outermost) we have at first the *philosophy foundation* of a work, the *research approach*, the *research strategy*, the *methodological choice*, the *time horizon* and finally *techniques and procedures*.

The research design includes a few choices for the research strategy, the methodology for data collection techniques, the procedures for analysis and time frame over which the research work is undertaken (see the three layers in Figure 26).

The centre of the research onion is about detailing *data collection and analysis*. Saunders, Lewis, and Thornihill (2009) consider it as ‘tactics’ rather than design and include clear explanation of the chosen data collection and data analysis procedures, each of which can be both quantitative and qualitative. Both research design and tactics explain the way research questions are addressed, which itself is influenced by the chosen research philosophy and approach (Saunders et al., 2019).

The stance of the layers in the research onion, from the outermost layer to the innermost layer contributes to ground the decisions around *data collection*, thus leading to a solid and credible approach of answering research questions. According to Crotty (1998), this is a way to show that a research work can be taken seriously.

In the following sections, each layer of the research onion is elaborated and contextualised relative to this particular research.

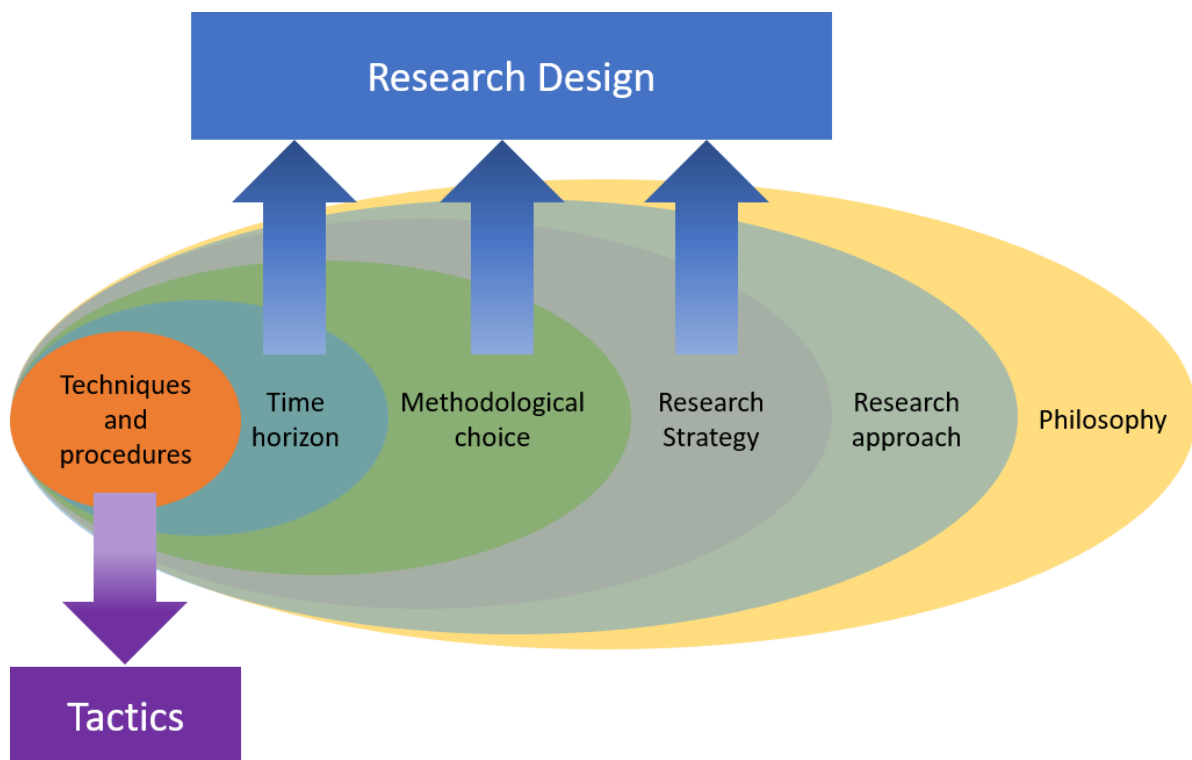


Figure 26. The “research onion”. Adapted from (Saunders, Lewis & Thornihill, 2009)

3.1 Philosophical Underpinning – Layer 1

The philosophical underpinning refers to the root concepts supporting the research which essentially includes a system of beliefs and assumptions about the development of knowledge in a particular field (Saunders et al., 2009). Such beliefs or assumptions help researchers to understand and investigate the phenomenon under study (Saunders et al., 2009; Mouton, 2001). In other words, understanding of research questions, used methods and interpretation of findings are all shaped by assumptions (Crotty 1998). According to Saunders, Lewis, and Thornihill (2009), a well-thought-out consistent set of assumptions underpins the methodological choice, research strategy along with data collection techniques and analysis procedures (see layers of the research onion in Figure 26).

The main philosophical assumptions can be grouped into three categories: *ontological*, *epistemological* and *axiological assumptions* (Saunders et al., 2009). The ontological assumption refers to the nature of the reality under investigation and what can be known about it. The epistemological assumption is concerned with the type of knowledge that is acceptable in the realm of study, and how to communicate it to the world. The *axiological assumption* is about the ways and to what extent in which the researcher's values influence the research process. To better clarify the role of each type of assumption, Table 1 not only summarizes their purposes but also reports some of the typical questions with answers that aid in a better comprehension from a philosophical stance:

- What is the nature of reality? (*ontology*);
- What is considered acceptable knowledge? (*epistemology*);
- Should the research be morally neutral, or should it be allowed to be influenced by one's values? (*axiology*).

Along with each type of philosophical assumption there are two positions one may choose to take: objectivism or subjectivism. Without digging much into the detailed application of each of the two positions with respect to individual philosophical assumptions, the two contrasting positions can be described as follows (Saunders et al., 2009):

- objectivism holds that social and physical phenomena (i.e. social reality) exists independently of individual's view;
- subjectivism holds that social reality is made from perceptions and consequent actions of individuals.

The most dominant philosophical paradigms (foundations) in the field of Information Systems are known as *positivism*, *interpretivism*, *constructivism* and *pragmatism* (Gregg et al., 2001; Deng & Ji, 2018). Each paradigm is characterized by a specific stance in relation to the ontological, epistemological and axiological assumptions, including an objectivism or subjectivism position. The choice of a paradigm, thereby, influences the way research is carried out, i.e. the way research queries are tackled (Saunders et al., 2009).

The set of philosophical assumptions that most suit this research work is the *pragmatism* paradigm, because it focuses on a specific problem for which a practical solution could be provided. In particular, the main research investigation should be solved by constructing a novel artefact, which makes a significant contribution into an application environment.

The need of a practical research approach recalls the *epistemological stance* of pragmatism, which is "knowing through making". Unlike other paradigms, in this epistemological stance, the validity of a concept or hypothesis is dependent not only on the *truthfulness*, but also on its *usefulness* in practical scenarios.

In line with the above, Lee and Nickerson (2010) define pragmatism as a school of thought where an idea can be better understood in terms of its practical consequences rather than its theoretical descriptions. Hence, we arrive at the *ontological stance* of pragmatism, where reality matters as a practical consequence of ideas.

From an *axiological* point of view, pragmatism recognizes the constructive role of researchers in the research process, which, according to Deng and Ji (2018), is analogous to the value of designers in the design process.

Both “focus on usefulness” and “constructive role of researchers” make pragmatism the most suitable paradigm to serve design science research (Deng & Ji, 2018; Hevner, 2007), which in turn, itself focuses on creation. The design science research will be elaborated later in this chapter along with research strategy.

The *ontological*, *epistemological* and *axiological* stances of pragmatism are summarized in the last row of Table 1. Each of these philosophical stances is below, further elaborated using the relevant aspects proposed in (Goldkuhl, 2011) (see the aspects also in Table 2).

Table 1. Ontological, epistemological and axiological assumption. Adapted from (Saunders et al., 2019)

	Ontological assumption Nature of reality or being	Epistemological assumption What constitutes acceptable knowledge	Axiological assumption Role of values in research
Purpose	Shaping the way in which the researcher sees and studies research objects (e.g. artefacts). The stance influences the focal point in a research project.	Identifying what can be considered as acceptable, valid and legitimate knowledge. The stance influences the choice of research methods, e.g. elicitation of insights rather relies on qualitative methods than quantitative.	Identifying the values of researcher and (if any) research participants. The stance influences the choice of data collection techniques, e.g. valuing data collected by personal interaction may lead to face-to-face interviews.
Questions	What is the nature of reality? What is the world like?	How can we know what we know? What is considered acceptable knowledge? What constitutes good-quality data? What kind of contributions can be made to knowledge?	What is the role of moral values in research? Should we try to be morally neutral with our re-search, or should we allow our values to shape our research? How should we deal with values of research participants?
Pragmatism	“Reality” is the practical consequence of ideas. External, multiple, views are chosen to enable best answering of research question. Flux of processes, experiences and practices	Practical meaning of knowledge in specific contexts. ‘True’ theories and knowledge are those that enable successful action to be taken. Either or both observable phenomena and subjective meanings can provide acceptable knowledge. Focuses on problems, practices and relevance. Integrating different perspectives to help interpret the data. Problem solving and informed future practice as a contribution.	Value-driven research. Research initiated and sustained by researcher’s doubts and beliefs. Values play a large role in interpreting results: the researcher can adopt either or both objective and subjective points of view.

In adopting the pragmatic assumption, the role of the researcher is of promoting changes (Haack, 1976). Simon (1996), in his seminal work, asserts that changing existing situations into preferred ones is the main aim behind all courses of action. Both actions and change characterize the ontological essence of the pragmatism (Goldkuhl, 2011). This, together with the interplay between action and knowledge, allow the researcher to intervene into the knowledge world where it can be applied. The purpose is to create utility (truth of this paradigm) by means of either creating novel artefacts or changes in organizations which are again related to the research approaches, namely design research (DR) and action research (AR) respectively. Both approaches provide constructive knowledge, which is the type of

acceptable knowledge in the pragmatic assumption, i.e. in epistemological aspect. The pragmatist researcher is oriented towards explaining and understanding prospective, prescriptive and normative aspects of knowledge. Based on these, guidelines, exhibit values and possible suggestions for practical improvements of the existing are provided. In other words, knowledge is created, so it can be practiced and is proven useful for action. Knowledge is constructed via inquiry, which is the mode of investigation sought to in some cases, in order to control the change in reality. The investigation is conducted by adopting methods, techniques and procedures to obtain and analyse data (Goles & Hirschheim, 2000).

Table 2. Aspects of the pragmatism. Adapted from (Goldkuhl, 2011)

<i>Empirical focus</i>	Actions and changes
<i>Type of knowledge</i>	Constructive knowledge
<i>Role of knowledge</i>	Useful for action
<i>Type of investigation</i>	Inquiry
<i>Data generation</i>	Through assessment and intervention
<i>Role of researcher</i>	Engaged in change

3.2 Inductive Research Approach – Layer 2

The second layer (starting from the outermost one) of the research onion proposed in (Saunders et al., 2009) deals with the research approach. The research approach is concerned with the use of theory, which can either be for testing or for building purposes. Although theory testing is associated to a deductive research approach, theory building is portrayed with an inductive approach (Saunders et al., 2009). The choice of one or the other steers the design of a research in aspects related to data collection techniques and analysis procedures as well as data sources and interpretation of findings.

This research work embraces an inductive approach as it aims to build a theory (i.e. a new artefact) rather than testing an existing one. In the following, the inductive research approach is first explained and then is described how the different aspects of the approach are relevant to this research.

Keeping in line with Saunders et al. (2019), inductive research starts with understanding the nature of a problem by collecting data. It is then the task of the researcher to extract meaning from the collected data through analysis. During this phase, patterns and regularities are, thereby, identified on the basis of cause-effect relationships. Based on the findings, a tentative hypothesis is then formulated; leading to a general conclusion and theories (e.g. a conceptual framework) finally completes the study. In contrast to the counterpart deductive approach, the inductive one is used for exploratory studies. According to Robson (2002), an exploratory study is useful to clarify the understanding of a problem by means of seeking new insights, asking questions, and assessing existing phenomena in a new light. Hence, reasoning is applied to produce theory, i.e. theory follows data and not the other way around, which would rather be the deductive approach. Since the reasoning is likely to be about the context of a subject under study, Saunders et al. (2019) suggest that a small sample of subjects is more appropriate in inductive approach, as compared to a large subject sample in deductive approach. The same authors add that “researchers in this tradition are more likely to work with qualitative data and use a variety of methods to collect these data in order to establish different views of phenomena” (Saunders et al. 2019).

Table 3 contains the described characteristics of an inductive research approach grouped by logic, generalisability, use of data and theory.

Table 3. Characteristics of an inductive research approach. Adapted from (Saunders et al., 2019).

Inductive Research Approach	
Logic	Generating untested conclusions using known premises.
Generalisability	Generalising from the specific to the general.
Use of data	Using of collected data for exploring a phenomenon, identifying themes and patterns and creating an artefact.
Theory	Generating and building a theory.

As depicted in Figure 27, in this research the observation bit is not only performed by two real-world cases of domain-specific adaptations of modelling languages but also by interviewing modelling experts. Data from these two research activities are analysed to improve understanding of problems and needs in domain-specific adaptations of modelling languages, respectively (see the pattern phase in Figure 27). The nature of the collected data is, thereby, qualitative. Once the patterns are identified, a set of design requirements is populated which is then used in the tentative hypothesis to conceive the new approach. Finally, a theory is devised in the form of a new approach for domain-specific adaptations of modelling languages.

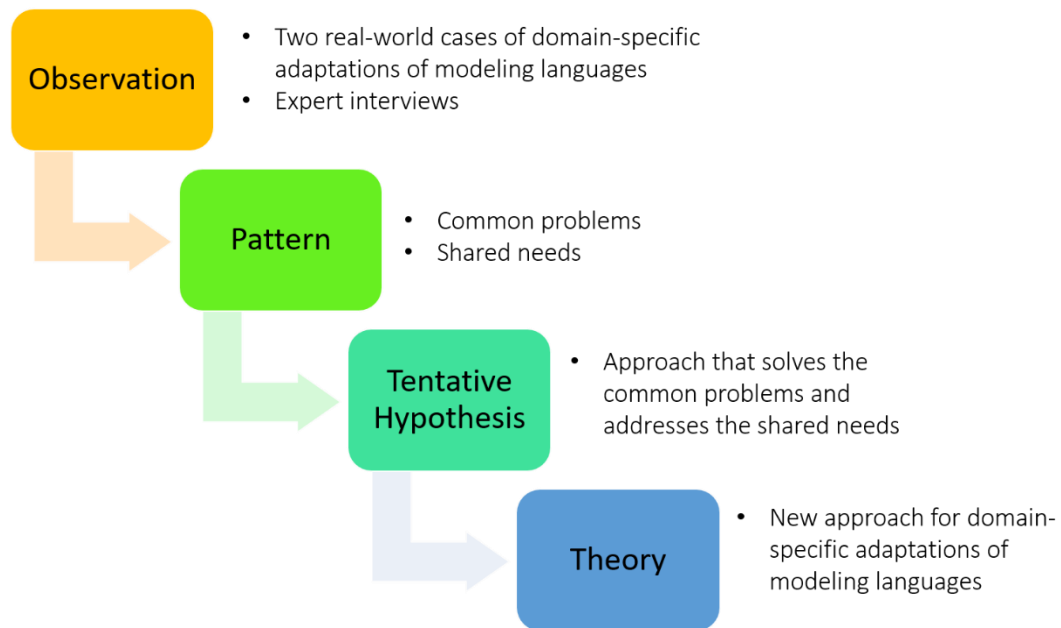


Figure 27 The adopted research approach. Adapted from (Saunders et al., 2009)

3.3 Research Strategy – Layer 3

According to Saunders et al. (2019) there is no research strategy that is more important than others. The question rather is, whether the chosen research strategy enables to answer the research question(s), and thus meets the research objectives. For this purpose, more than one research strategies can be adopted and combined (Saunders et al., 2019). Possible research strategies are: *experiment*, *survey*, *action research*, *case study*, *grounded theory*, *ethnography* and *archival research* (Saunders et al., 2019) and *design science research* (Hevner et al., 2004).

The choice of an appropriate research strategy is influenced by the philosophical underpinnings (Easterby-Smith et al., 2008; Saunders et al., 2019). As anticipated in Section 3, pragmatism is considered as the most suitable paradigm to serve the research strategy design science research (DSR). Deng and Ji (2018) claim that pragmatism is the only philosophy able to fully address the unique requirements of DSR, which focuses on creation. Additionally, Hevner (2007) states that “design science research is essentially pragmatic in nature due to its emphasis on relevance; making a clear contribution into the application environment”.

The *design science research* is the *main research strategy* followed in this research. Additionally, a *case study research* strategy is adopted to further increase the awareness of the research problem, which corresponds to the initial phase of design science research.

In the following, the theoretical background of both strategies the design research and the case study are first introduced. Next, the two strategies are contextualised in this research work, further describing how they support answering to the research questions creating an interlink between one another.

3.3.1 Design Science Research

Design science research (DSR) is a research strategy that focuses primarily on problem-solving (March & Storey, 2008). In contrast to other sciences, like natural science and social science, whose aim is understanding reality, design science has its roots in engineering and the sciences

of the artificial (Simon, 1996). Hence, design science endeavours to build artefacts that serve human purposes (March & Smith, 1995). In other words, research in design science transforms the “present situation” to the “desired situation” by constructing and evaluating artefacts that address organizational problems.

Now, what do we understand by the term ‘artefacts’? Hevner et al. (2004) define artefacts as “innovations that define the ideas, practices, technical capabilities, and products through which the analysis, design, implementation, and use of information systems can be effectively and efficiently accomplished”. An artefact is an innovative product that is derived through a design process, i.e. a sequence of expert activities (Hevner et al., 2004). According to Deng and Ji (2018) “Information Systems is one area that exhibits increasing adoption of Design Science as an epistemological paradigm of the advancement of knowledge”.

To increase high quality design science research in Information Systems, Hevner (2007) proposed a conceptual framework containing three design science research cycles: (1) the *design cycle* (2) the *relevance cycle*, (3) the *rigor cycle* (Figure 28).

The *design cycle* (centre of the framework shown in Figure 28) is essentially where the artefact is built and evaluated iteratively. The iteration continues until a satisfactory design is achieved catering to requirements (Simon, 1996). As Hevner (2007) states, requirements for the construction of the artefact are derived from the relevance and design cycle while evaluation theories and methods are from the rigor cycle. The cardinal industrious work takes place in the design cycle, paying adequate significance to both relevance and rigor cycles throughout the design process. (Hevner, 2007).

The *relevance cycle* aims to improve the environment by introducing innovative artefacts. The environment (left extreme block of the framework shown in Figure 28) can be defined as the problem space in which the application domain resides. An application domain consists of people along with organizational and technical systems that interact with each other, working towards a goal (Hevner, 2007). Business needs or problems raise from an application domain and are related to existing technology infrastructure, applications, communication architectures, and development capabilities (Hevner *et al.*, 2004). Such business needs or problems are perceived by people within the organization, from whom they can be captured by the researcher. The latter should, thereby, conduct research activities that address business needs or problems in order to assure the relevance of the research. Hevner (2007) suggests that a relevance cycle initiates the design science research with an application context to provide requirements for the research (e.g. business needs or problems) and define acceptance criteria for the evaluation of the research results. The cycle then, returns the output from the design science research into the environment for further studies and evaluation in the application domain, eventually coming to an end. From the field testing, if limitations (on the utility of the produced artefact) arise, the relevance cycle will iterate again, starting with the new requirements from the environment.

The *rigor cycle* ensures that the produced artefact is innovative and rigorously designed from a theoretical point of view. As shown by the block on the right extreme of Figure 28, the knowledge base provides not only the theoretical foundations and methodologies, but also the experiences and expertise that define a state-of-the-art application domain of the research including the existing artefacts and processes found in the application domain (Hevner, 2007). The selection and application of the appropriate theories and methods for building and evaluating the artefact is solely determined by the researcher’s skills. The rigor cycle concludes by adding the extended theories, methods and experiences that were gained while conducting the design science research back in the knowledge base.

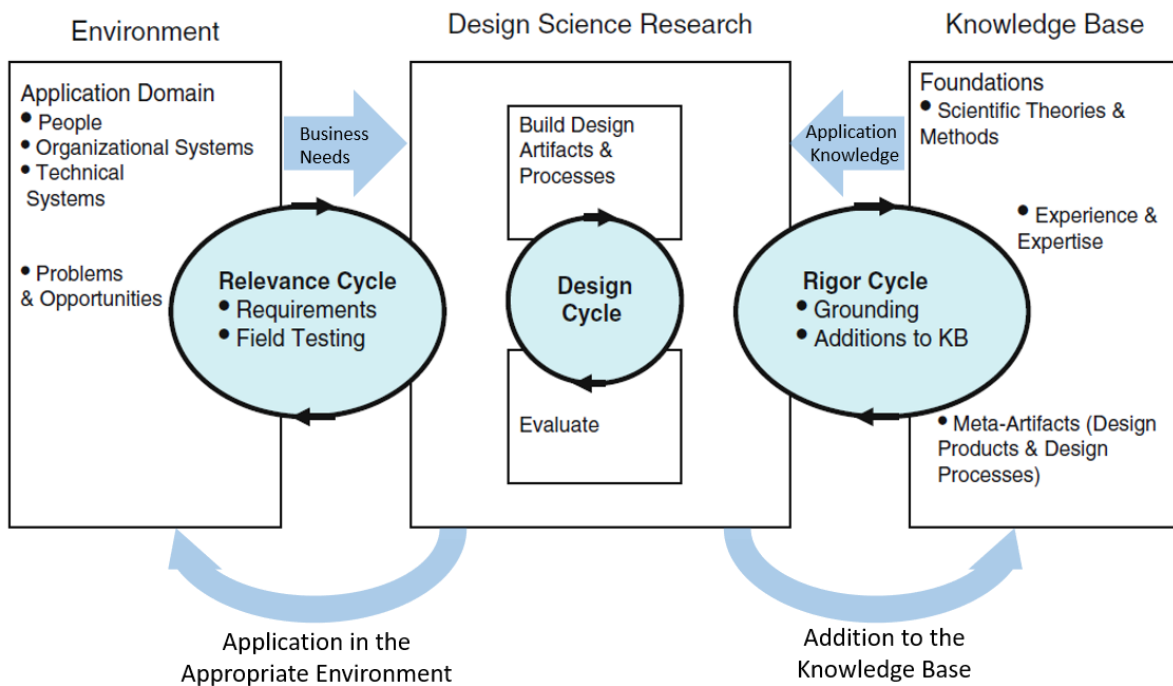


Figure 28. Design science research conceptual framework and cycles. Adapted from (Hevner, 2007)

Vaishnavi and Kuechler (2004) elaborate on the design cycle by proposing a general design cycle (GDC), which foresees five sequential research phases, namely: *Awareness of Problem*, *Suggestion*, *Development*, *Evaluation* and *Conclusion* (see Figure 29). The two cycles for relevance and rigor are engaged in each of the phases.

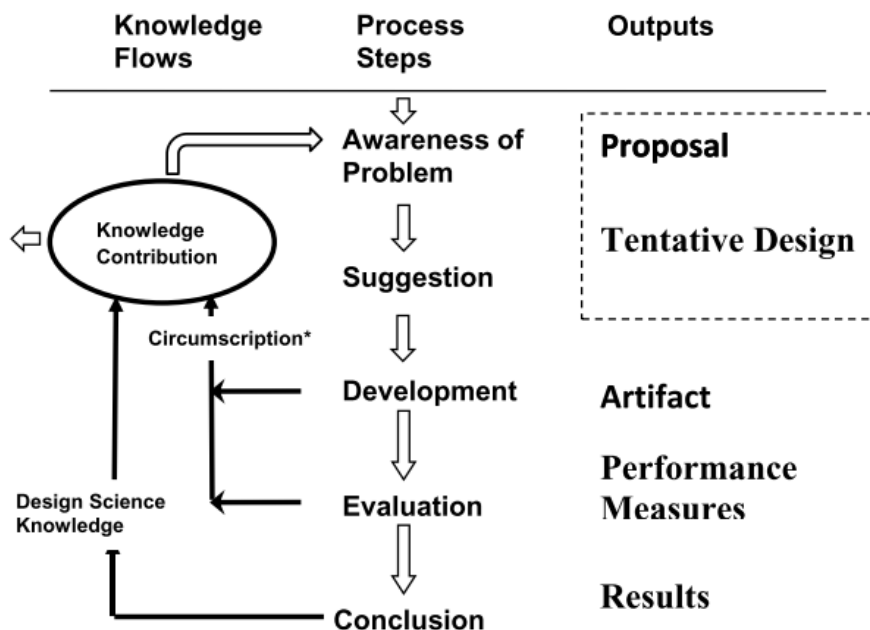


Figure 29. Reasoning in the general design cycle (GDC) (Vaishnavi and Kuechler, 2004)

- In the *Awareness of Problem* phase, a research problem is raised and may originate from multiple sources including the above-mentioned environment and knowledge base. Thus,

in this phase, both the relevance and rigor cycles are used to input the design cycle. In this phase, additional research strategies can be adopted to increase the understanding of a problem. For instance, with an inductive approach, the case study strategy would be best way to ensure deep understanding, whereas with a deductive approach the survey strategy would be more appropriate (Saunders et al., 2019).

- In the *Suggestion phase*, the problem identified in the previous phase is tackled by proposing a tentative design of the new artefact. An artefact must be in the form of a construct, a model, a method or an instantiation (Hevner et al., 2004). Its tentative design is drawn from the knowledge acquired. According to Vaishnavi and Kuechler (2004), suggestion is essentially a creative step.
- In the *Development phase*, the proposed design of the artefact is further refined and implemented into an IT artefact. An IT artefact is a tangible socio-technological instance of the suggested theoretically grounded artefact (see suggestion phase), which according to Koppenhagen, et al. (2012), can be experienced, discussed, tested, evaluated, changed, improved and extended. The relevance is provided by the real-world socio-technological context in which the artefact is implemented (e.g. suggestion about the technological adoption). The fact that the IT artefact derives from a theoretically grounded construct (model, method or an instantiation) assures rigor. This implies to make the relations between the artefact and its instantiation explicit and traceable (Koppenhagen et al., 2012).
- In the *Evaluation phase* the artefact is evaluated. March and Storey (2008) state that “the representation of design problems and the generation and evaluation of design solutions are the major tasks in design science research”. In line with this claim, Iivari (2007) further stresses the importance of the Evaluation phase by asserting that it is “the essence of Information Systems as design science lies in the scientific evaluation of artefacts”. An artefact evaluation aspires to determine the progress achieved by designing, constructing, or using an artefact in relation to the identified problem and the design objectives (March & Smith 1995; Aier & Fischer, 2011; Sonnenberg & vom Brocke, 2012). To achieve improved rigor in DSR, an evaluation strategy can be proposed wherein evaluation criteria is made explicit. The formation of an evaluation strategy in DSR can be underpinned by existing strategic frameworks such as the one proposed in (Pries-Heje, Baskerville & Venable, 2008). To assure relevance, we should consider naturalistic evaluations, which are critical to prove the artefact’s utility for practice (Hevner et al., 2004). This type of evaluation considers aspects of real-world environment, which aim to accomplish real tasks in real settings, which can include real users, real systems and/or real problems (Pries-Heje et al., 2008). According to Sonnenberg and vom Brocke (2012), such evaluation types could incorporate the organizational context partially or entirely. Artefact evaluation results are then used to iterate the (above explained) design cycle, probably giving rise to new requirements for the artefact. When the evaluation results are satisfactory with respect to the imposed criteria, the evaluation phase ends.
- The *Conclusion phase* is the end of the research cycle and of the research effort (Vaishnavi & Kuechler, 2004). The research results are considered as good enough and are consolidated together with the knowledge gained and produced. Hence, contributions to the knowledge of body and to the practice should to be made explicit. This aims to assure rigor and relevance of the research, respectively (Hevner et al., 2004). To demonstrate the produced high quality design science research, the results can be compared to the seven design science research guidelines proposed in (Hevner et al., 2004). Finally, limitations and future research directions should be addressed. Future directions are particularly

important as the truth builds only on previous discoveries (Keith, Vitasek, Manrodt, & Kling, 2016).

3.3.2 Design Science Research in this Research Work

This research work encompasses the just introduced research strategy, *design science research* (DSR). The DSR methodology adopted in this research is shaped from the two works of Hevner *et al.*, (2004) and Vaishnavi and Kuechler, (2004) and is graphically depicted in Figure 30. The five DSR phases of the generic design cycle (GDC) are shown in the centre of the figure while the environment and the knowledge base are depicted on the left and right-hand side of the DSR phases, respectively. The outputs of the five DSR phases are shown in a separate column (Right extreme of Figure 30). This chapter further discusses each DSR phase elaborately, pointing the output as well as the relation with both the environment and the knowledge base.

Awareness of Problem: This phase starts by identifying the research problem, which includes the research problem description (Section 1.1), research objectives and research questions (Section 1.2). The literature review is then presented in Chapter 2 and it backs up the research problem. Next, the first research question RQ1 “*What are the problems that hinder agility in domain-specific adaptations of modelling languages?*” is addressed, which aims to raise the awareness of problems in the environment. The case study research strategy supports in tackling this research question. Two case studies were executed, which address two different application domains: the patient transferal management (i.e. a digital healthcare application domain) and business process as a service (i.e. a Cloud Computing application domain) (see Sections 4.1 and 4.2, respectively). While the former was conceived during the Swiss research project Patient-Radar¹⁵ (Reimer & Laurenzi, 2014), the latter was structured during the European research project CloudSocket¹⁶ (Woitsch & Utz, 2015). Both the above-mentioned case studies not only facilitated the data collection as experts were available for workshops and interviews but also shared material like real-world use cases and relevant documents from industry partners among the project participants. In both case studies two domain-specific modelling languages (DSMLs) were developed by adopting the Agile Modelling Method Engineering (AMME) lifecycle (Karagiannis 2015; Karagiannis 2018; Bork *et al.*, 2019). The choice of the AMME lifecycle is driven by the findings from the literature investigation of methodologies for meta-modelling. While the two case studies assure relevance, the literature investigation assures rigor (see both relevance and rigor cycles in Figure 30). Since the AMME lifecycle was already broadly adopted to create modelling methods, tools and languages in the meta-modelling toolkit ADOxx (see (Karagiannis, *et al.* 2016)), the latter was chosen to create the two DSMLs. Both the case study strategy and its design (which underpins the answer for the two research questions) are elaborated in the Sub-sections 3.3.3 and 3.3.4. From the analysis of the two cases, a list of problems hindering agility were derived. The problems were then compared to literature findings to derive the two main challenges to address (Section 4.3).

The second research question RQ2 “*What are the needs for domain-specific adaptations of enterprise modelling languages?*” was also addressed in this phase. To address this question, data were further collected by modelling expert interviews (Sub-section 4.4.2) were conducted. Findings were then consolidated and analysed to derive the first set of requirements to address by the new approach of this research work (Sub-section 4.4.5).

Suggestion: In the suggestion phase the third research question RQ3 “*How can agility be fostered when performing domain-specific adaptations of modelling languages?*” is addressed.

¹⁵<https://www.fhsg.ch/en/projects/project/patient-radar-226/>

¹⁶ <https://site.cloudsocket.eu/>

For this, the challenges and requirements derived in the previous phase were addressed, to provide a tentative design of the artefact. While designing the agile meta-modelling artefact (Section 5) additional literature inquiry was conducted about the semantic specifications. This reflects the use of the rigor cycle also in the suggestion phase, which led to additional requirements (see Sub-section 5.1.5). As a result, a set of operators for an agile meta-modelling were conceived, which addressed all the requirements (Section 5.1.6). The relevance cycle in this case was used once more to validate the set operators through interviews with modelling experts. Next, a critical reflection on the tentative design led to list of limitations (Sub-section 5.1.6.3). The artefact was then extended to overcome these limitations, by grounding the agile meta-modelling with an ontology-based approach (Section 5.2). The rigor cycle was iterated again to inject ontology approach experiences, such as Semantic Lifting (5.2.1) and ontology-based meta-modelling (5.3.1). Since the latter resulted to be more appropriate for the agile approach, artefacts were built upon the ontology-based meta-modelling. Thus, an ontology architecture is proposed (Sub-section 5.3.2), which is followed by a set of semantic rules (Sub-section 5.3.3). The semantic rules were conceived by following the methodology of Grüniger & Fox (1995), meaning that the rigor cycle was implemented in this phase, one more time. At the end of the phase, the delivered artefact was an agile, ontology-aided meta-modelling.

Development: This phase answers the last research question RQ4 “*How can the agile approach for domain-specific adaptations that preserves seamless consistency between the graphical and the machine interpretable representation be automated?*”. For this, the artefact suggested in the previous phase is instantiated and implemented into the so-called AOAME (Agile and Ontology-Aided Meta-modelling Environment). The instantiation assured the rigor of the artefact. Functionalities in AOAME are thereby implemented to allow automation of the on-the-fly domain-specific adaptations while preserving consistency between the human and the machine-interpretable representations (Section 6.3). The Model-View-Controller design pattern was adopted to implement the approach (Sub-section 6.1.3). This design pattern belongs to the practice of web-based software development, which comes from the environment (left-hand side of Figure 30). Hence, the relevance cycle was used in this phase.

Evaluation: The evaluation phase starts by reporting literature about evaluation strategies in design science research (Section 7), which is the further demonstration on the use of the rigor cycle. Based on the findings, a suitable research strategy for this research work is built (Section 7.2.). From the research strategy three main evaluation criteria were derived:

1. *Correct design of the artefact:* The extent to which requirements are satisfied by the implemented approach AOAME.
2. *Operationability of the artefact:* The ability of the approach to preserve consistency between the human- and the machine interpretable representation while performing on-the-fly domain-specific adaptations of modelling languages.
3. *Generality of the artefact:* The ability of the approach, characteristic to this research, to be applied in different application domains.

For all the three criteria the evaluation was conducted qualitatively and AOAME was used as a means of evaluation. The evaluation with respect to the first criteria is done against all the eight design requirements elicited during the *awareness of problem* and *suggestion phases*. This evaluation is described in Section 7.3. The operationability and generality of the approach was demonstrated by the implementation of real-world cases in AOAME, addressing different application domains (see Section 7.4).

The evaluation phase triggered the design cycle (see between development and evaluation phase in Figure 30), which in turn refined the artefact iteratively. The improvements are discussed in the conclusive section of the evaluation phase (Section 7.5).

Conclusion: In this conclusive phase, a comprehensive summary about how the research questions were answered is provided in Section 8.1. The contributions of the research are reported as follows (Section 8.2). The main artefact is presented together with its sub-artefacts. The contributions to the practice and the body of knowledge are then presented. The two contributions correspond to the two grey arrows in Figure 30 that from the design science research go to the environment and the body of knowledge, respectively. To manifest that the research adhere to a high quality design science research, aspects of this research work are compared to the seven guidelines proposed in (Hevner et al., 2004) (Section 8.3). Limitations of the research are presented together with the possible future work (Section 8.4).

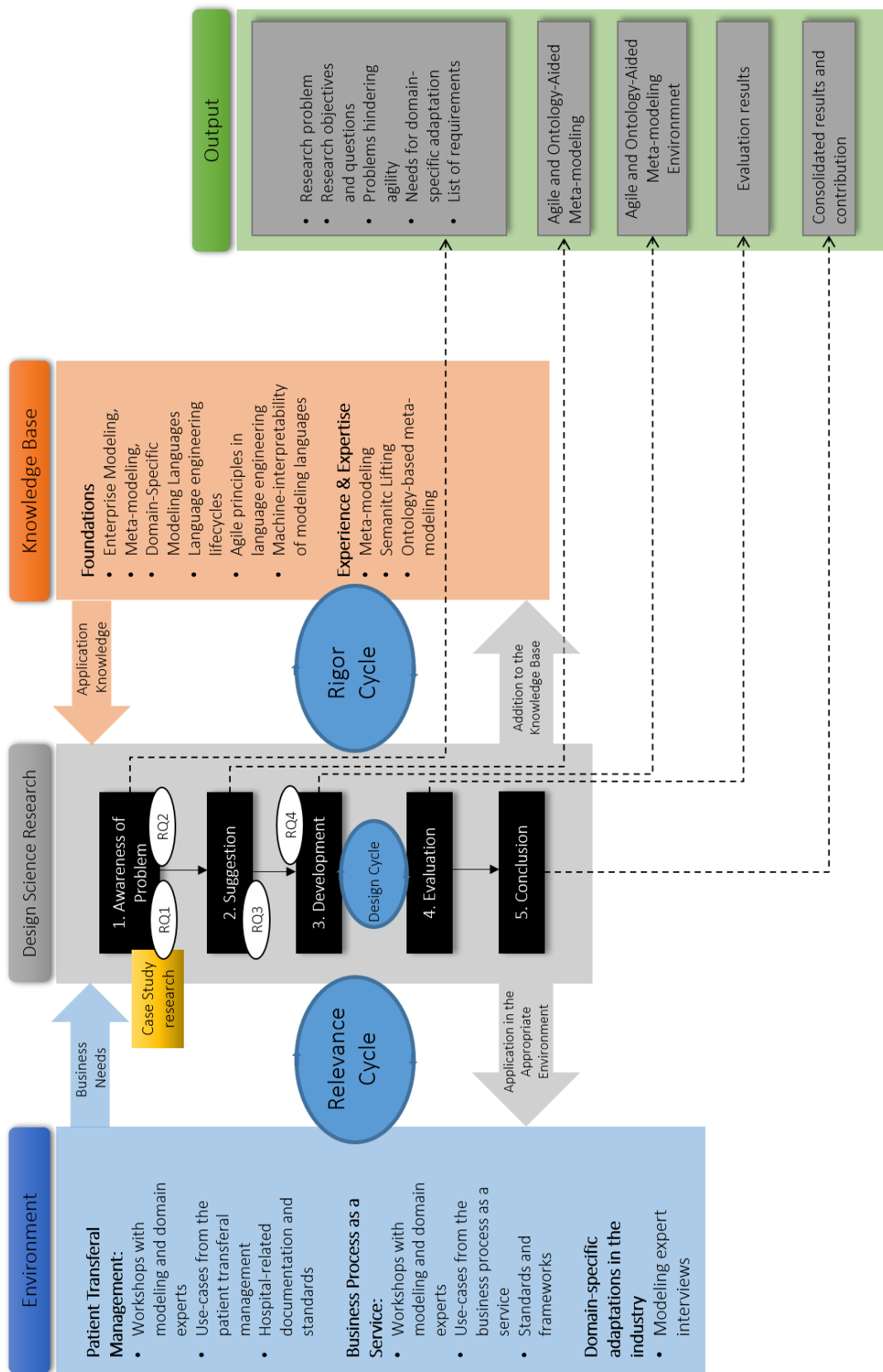


Figure 30. Design Science Research (DSR) methodology applied in this research work. Adapted from Vaishnavi and Kuechler (2004) and DSR cycles from Hevner (2007)

3.3.3 Case Study Research

According to Yin (2003), the case study research is an empirical inquiry that investigates the case or cases by addressing the “how” or “why” questions concerning the phenomenon of interest. Robson (2002) defines the case study as “a strategy for doing research which involves an empirical investigation of a particular contemporary phenomenon within its real life context using multiple sources of evidence”. This definition is in line with what Yin (2003) defines as a case, i.e. “a contemporary phenomenon within its real life context, especially when the boundaries between a phenomenon and context are not clear and the researcher has little control over the phenomenon and context”.

The case study strategy allows gaining a rich understanding of the context of the research (Yin, 2003), and is often used in explanatory and exploratory research (Saunders et al., 2019). Yin (2003) suggests adopting multiple cases over a single case. This contributes to the generalisation of the findings. Yin (2003) also defines the unit of analysis for a case study strategy: holistic vs. embedded. A holistic design requires one unit of analysis, whereas an embedded design requires multiple units of analysis.

Yazan (2015), in a comparison among most influential case study strategies in social science research, stresses the different positions related to design approaches. Whereas Yin (2003) suggests a tight and structured design for case study strategies, Stake (1995) promotes a flexible design allowing the researchers to make changes midway of conducting the research. According to Stake (1995), in an initial design, pointing out issues and issue questions are sufficient, which then to research questions. In contrast, Yin (2003) advises to rigorously design a case study comprising of the following five proposed components:

1. question(s): represent the queries that the case should address;
2. proposition(s): it highlights the issues that should be examined within the scope of the case study;
3. unit(s) of analysis: it specifies what should be analysed elucidating what the case is; it is related to the way the research questions are defined;
4. logic linking the data to the propositions: anticipates the possible steps involved in the data analysis (e.g. pattern matching);
5. criteria for interpreting the findings: different patterns are sufficiently contrasting such that findings can be interpreted alluding to at least two rival propositions.

Yin (2003) suggests that the five components are cohesive to and consistent amongst each other. Before commencing data collection, Yin (2003) suggests to review the relevant literature and include theoretical propositions regarding the case under study.

The employed data collection techniques to create a case may vary or merge, e.g. interviews, observation, documentation, questionnaires and physical artefacts (Saunders et al., 2019; Yin; 2003). The analysis of data then “consists of examining, categorizing, tabulating, testing, or otherwise recombining both quantitative and qualitative evidence to address the initial propositions of a study” Yin (2003). According to the author, there are five dominant techniques for data analysis: pattern matching, explanation building, time-series analysis, program logic models, and cross-case synthesis.

Finally, in order to maintain the quality of the investigation, Yin (2003) suggests to measure the case with respect to four criteria namely:

1. Construct validity: through the triangulation of multiple sources of evidence, chains of evidence, and member checking. Similarly, in (Saunders *et al.*, 2019) the authors stress

the triangulation of multiple sources of data, which aim at ensuring credibility of data collection.

2. Internal validity: using one of the above-mentioned techniques for data analysis (e.g.: explanation building or pattern matching). According to Merriam (1998), the internal validity can be improvised by triangulation, member checks, long-term observation, peer examination and participatory research.
3. External validity: through analytic generalization. As mentioned above, the adoption of multiple cases contributes to the generalization.
4. Reliability: through case study protocols and databases. Reliability can also be ensured for strategies for triangulation such as data source triangulation (Stake 1995; Yazan 2015).

3.3.4 Case Study Research in this Research Work

In this sub-section, the design of the case study strategy adopted in this research is described. Following the suggestion of Yin (2003), our proposed case study strategy includes the creation of two cases (multiple case) as well as focuses on one unit of analysis (holistic design). The five components suggested by Yin (2003) are applied in the design of the two cases:

1. Questions: the first research question (RQ1) is addressed in both cases: What are the problems that hinder agility in domain-specific adaptations of enterprise modelling languages?
2. *Proposition*: as suggested by Yin (2003), relevant literature has been reviewed and is described in Chapter 2. Each case created a DSML by following the AMME Lifecycle. Findings from the literature review have shown that the AMME Lifecycle is the most suitable methodology. Namely,
 - it allows creating DSML through domain-specific adaptations,
 - it is wide-spread, and
 - it embraces agile principles. Research findings have also shown that a DSML have not only the purpose of documenting knowledge but also automating knowledge. Therefore, one case fell within the field of Knowledge Management (KM) while the second one is extended to the field of Knowledge Engineering (KE).
3. *Unit of analysis*: is the only unit of analysis and this is the language engineering lifecycle, through domain-specific adaptations. Therefore, in order to raise the problems hindering agility, the two cases are analysed by focusing on the language engineering lifecycle of AMME.
4. *Logic linking the data to the propositions*: where explanation building and pattern matching are used as techniques for data analysis. The former is used to present the problems and to allude them to the data. Since a few activities of the creation of the two DSMLs overlap, pattern matching is used to identify such problems afflicting the creation of both DSMLs.
5. *Criteria for interpreting the findings*: that are consolidated and compared to the existing theoretical ones.

In order to ensure the quality of the investigation, the four criteria proposed by Yin (2003) are considered:

1. *Construct validity*: triangulation of data sources. Each case is constructed through workshops with modelling and domain experts, relevant documentation and real-world use cases from the respective application domains and literature review.
2. *Internal validity*: member checks and peer examination are used to ensure the internal validity of both cases. Members and peers are from the project members (CloudSocket for the BPaaS case and Patient-Radar for the Patient Transferal Management).
3. *External validity*: the external validity is ensured by the publications in conference proceeding made for each of the two cases. Additionally, the multiple case considerations ensure the generalization of the findings.
4. *Reliability*: the triangulation of different data sources ensures the reliability of each case.

3.4 Methodological Choice – Layers 4

This sub-section presents the last two layers of the research onion that (according to Saunders et al. (2019)) have the purpose of designing the research: (1) the methodological choice for both collecting and analyzing data and (2) the time horizon. Below, the methodological choice is described first.

The methodological choice refers to the way one may combine data collection techniques and data analysis procedures. One significant distinction in the collection techniques and analysis procedures is about their focus on either numeric (quantitative) or non-numeric data (qualitative). Any data collection technique or data analysis procedure that is quantitative generates or uses numerical data. An example is the survey technique from which results can be subject to statistical methods. In contrast, any qualitative data collection technique or analysis procedure generates or uses non-numerical data, i.e. words or phrases. An example is the interview technique from which results can be categorised. A qualitative research is, therefore, rather concerned with rich data including opinion, description and personal accounts.

According to Saunders et al. (2019), there are mainly three methodological choices that can be considered to answer each research question: *mono-method*, *mixed-method* and *multi-method* (see picture Figure 31). The mono-method refers to a single (qualitative or quantitative) data collection technique and corresponding analysis procedures. In contrast, both mixed-methods and multi-methods use more than one data collection techniques and analysis procedures. When more than one quantitative data collection techniques are adopted, the multi-methods design refers to *multi-method quantitative studies*. On the other hand, if the multiple data collection techniques are qualitative, the methodological choice is called *multi-method qualitative studies*. Moreover, in multi-method studies, the type of procedures used for data analysis (i.e. qualitative or quantitative) must correspond to the same type chosen for the data collection. This implies, a qualitative data collection cannot be associated with a quantitative analysis of data. It becomes rather the case of mixed-methods in which qualitative and quantitative techniques and procedures are mixed up.

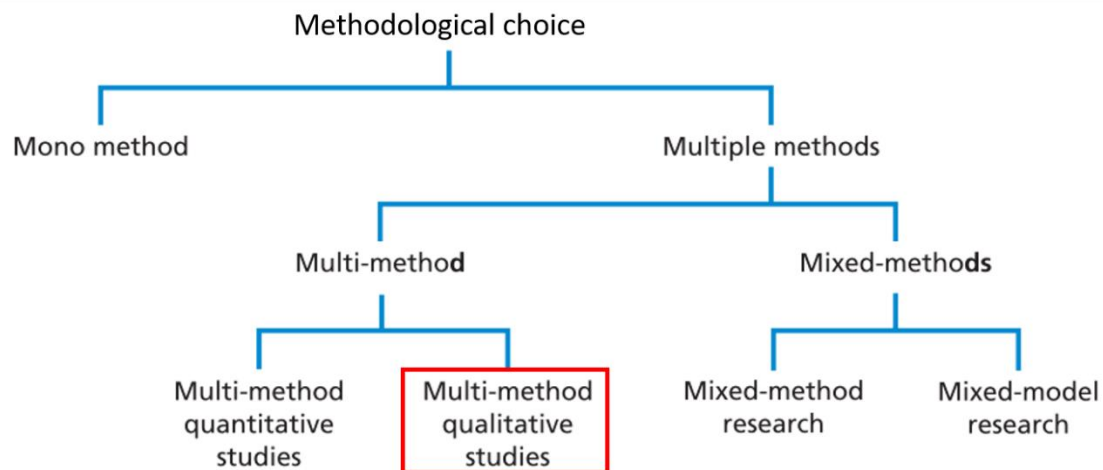


Figure 31. Methodological choices (adapted from Saunders et al. 2019)

Saunders et al. (2019) suggest that the philosophical underpinning of a research work provides a direction towards an appropriate methodological choice. In philosophies such as interpretivism and positivism the methodological choice is narrowed into either qualitative or quantitative techniques, respectively. This is different in pragmatism (the philosophical stance of this research – see Section 3) where the methodological choice is driven by the research problem and questions. The range of methods, thereby, varies from qualitative or quantitative; mono, mixed or multi methods.

According to Tashakkori and Teddie (2003), the adoption of multiple methods creates better opportunities to answer research question(s) than mono methods. The main advantage of the use of multiple data collection methods is that it increases the trust in the research findings (Tashakkori & Teddie, 2003). An examination of 232 social science articles conducted by Bryman (2006), skimmed down to the result that the adoption of multiple methods creates a wealth of data way above the researcher’s expectations.

Whereas quantitative multi-methods are more appropriate for deductive research approaches, qualitative multi-methods are typically used for inductive approaches and problem solving (Klein & Myers, 1999; Kaplan & Maxwell, 2005). Klein and Myers (1999) suggest qualitative methods to attain a deep understanding of the phenomenon under study.

As described in Section 3.2, this research is characterized by an inductive approach. Moreover, the main research question can be answered only after the following deep understandings are achieved:

- the way domain-specific adaptations are performed to create domain-specific modelling languages – from this understanding problems in the practice of domain-specific adaptations can be derived.
- the relevant domain-specific adaptations in industries – from this understanding needs for domain-specific adaptations can be derived.

Therefore, the methodological choice in this research falls into the *qualitative multi-methods*. In order to corroborate the research findings, a *triangulation approach* is adopted, in which more than two independent sources of data are considered as well as more than two data collection methods. According to Saunders, et al. (2019), a triangulation approach helps ensuring *credibility* of data, i.e. that “the data are telling you what you think they are telling you” (Saunders et al., 2019). The specification of the data collection techniques and data

analysis procedures belong to the “tactics” of the research onion (i.e. layer 6) and therefore are described in the next sub-section.

3.5 Time Horizon – Layer 5

The last aspect of the research design is concomitant with *time horizons*. According to Saunders, et al. (2019), time horizons can be distinguished between *cross-sectional* and *longitudinal approaches*. The cross-sectional studies refer to “a particular phenomenon (or phenomena) at a particular time” (Saunders et al., 2019). Most research projects are cross-sectional as they are time constrained. In contrast, longitudinal refers to the studies conducted over a long period of time in which the phenomenon (or phenomena) is investigated at multiple time points.

Since this research is about investigating a phenomenon within a constrained time period, it is essentially a *cross-sectional study*.

3.6 Data Collection and Data Analysis – Layer 6

As afore-mentioned, at the beginning of the chapter, the focal point is the way to answer research questions and what data are collected as well as how data are analysed. In the research onion of Saunders et al. (2019) the specifications of these aspects falls under the last layer. In the following, the focus is first on data collection techniques and then on analysis procedures.

Data can be distinguished between primary and secondary. For both the types a qualitative or quantitative method can be adopted. *Primary data* refers to data collected by the researcher himself or herself. Hox and Boeije (2005) assert that the primary data are “*collected for a specific research problem at hand, using procedures that fit the research problem best*”. On the other hand, the term *secondary data* refers to data that are collected by someone else and can have a different research goal, a different research question or, a different context, thus it is a data that already exists. Qualitative secondary data can be existing documents, transcripts or models.

Hox and Boeije (2005) argue that for secondary data with qualitative nature their easy access (i.e. lower costs and faster access) may come at the expenses of a difficult interpretation. On the other hand, although the collection of primary data is costly and time consuming, they are tailored to resolve the problem.

In this research work, primary and secondary data were collected qualitatively using multiple independent data sources as well as techniques. As already mentioned in the previous section, this is also known as a *triangulation* scientific approach, which helps acquiring a richer understanding of the phenomenon under study and it enhances the trustworthiness of data, i.e. credibility and validity (Saunders et al., 2019).

The triangulation occurs on two dimensions: *source triangulation* and *method triangulations*. Source triangulation refers to different data sources from different time horizons or different data capturing settings, while method triangulations refer to multiple data collection techniques (Patton, 2015).

In this work, the two triangulation dimensions were first employed to support the *case study research* (see Sub-section 3.3.4), and hence to answer the first research question RQ1: *What are the problems that hinder agility in domain-specific adaptations of enterprise modelling languages?*

The objective is to derive a list of problems that hinder agility when performing domain-specific adaptations on modelling languages as well as setting the challenges. For this, two DSMLs were created to increase understanding of how domain-specific adaptations are

performed. The following data collection aspects were therefore considered (see also right hand-side of Figure 32):

- *Source triangulation*: (1) project partners with domain expertise; (2) project-related documentation (including real-world use cases) from both projects CloudSocket and Patient-Radar; (4) theories.
- *Triangulation methods*: (1) interviews (2) focus group (3) literature review.

The two triangulation dimensions are further employed to spread the awareness of problem phase of the *design science research*. In particular, the data collection should support the objective of conceiving a set of requirements to be fulfilled by the new artefact (see left hand-side of Figure 32). To ensure credibility each requirement was crossed-supported by the triangulation among *problems* (identified by analysing the two cases), *needs* (identified by interviewing the modelling experts) and *literature*. Since the two cases are independent from each other they can be considered as two separate data sources. The needs for domain-specific adaptations are identified by answering to RQ2: *What are the needs for domain-specific adaptations of enterprise modelling languages?*

As a result, the following data collection aspects were considered (see also Figure 32):

- *Source triangulation*: (1) Patient Transferal Management application domain, (2) Business Process as a Service application domain, (3) modelling experts, and (4) theories.
- *Triangulation methods*: (1) case study; (2) interviews; and (3) literature review.

The details about the source triangulation adopted in the case study research and the design science research are provided in Sub-sections 3.6.1 and 3.6.2, respectively.

Three of the four considered data collection techniques (1. literature investigation, 2. expert interviews and 3. focus group) were considered as the three principal ways of conducting exploratory research (Saunders et al., 2019). Respectively, in Sub-sections 3.6.3, 3.6.4.1 and 3.6.4.2 the three triangulation methods are further elaborated with theoretical background. The case study was already described in Sub-section 3.3.3.

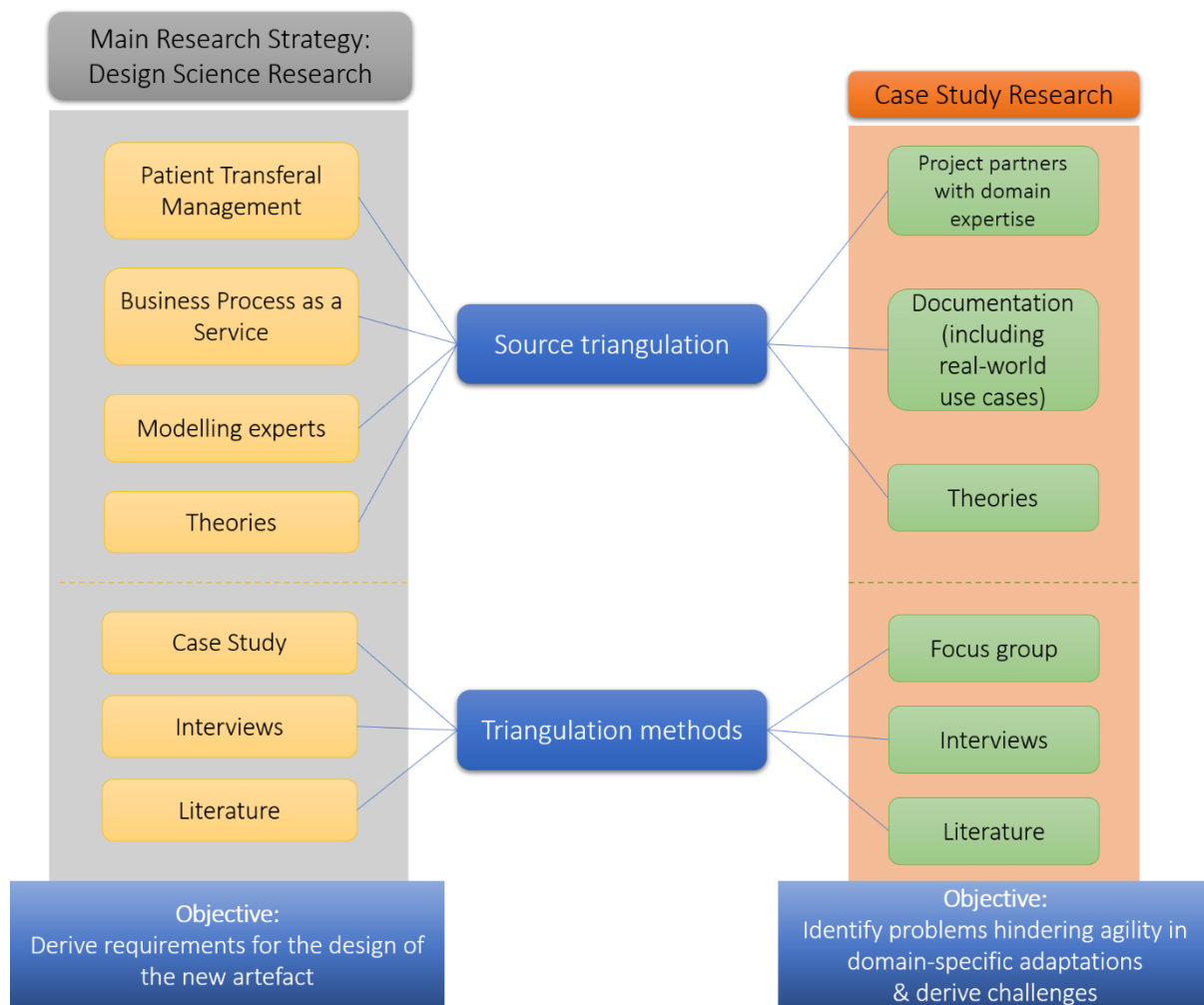


Figure 32. Source triangulation and method triangulation employed in this research

3.6.1 Source Triangulation in the Case Study Research

For the first case (i.e. Patient Transferal Management) the following data sources were considered:

- Partners of the project Patient-Radar were: physicians, transferal managers and process modelling experts. The project facilitated the interaction with the experts.
- Documentation:
 - o Project-related standards (secondary data) - DefReha© (Gli Ospedali Svizzeri, 2017), International Classification of Functioning, Disability and Health (ICF) standard (World Health Organization, 2016) and International Statistical Classification of Diseases and Related Health Problems¹⁷ (ICD-10).
 - o Documents developed during the project (primary data) are retrievable in Appendix A: Patient Transferal Management.

¹⁷ <http://apps.who.int/classifications/icd10/browse/2016/en>

- Related scientific work (secondary data) - see Sub-section 4.1.3.

For the second case (i.e. Business Process as a Service) the following data sources were considered:

- Partners of the project CloudSocket namely: business process managers, cloud computing experts and cloud brokers. The project facilitated the interaction with the experts.
- Documentation:
 - o Project-related standards (secondary data) - APQC Process Classification Framework (APQC 2014) and Cloud Service Level Agreement Standardisation Guidelines (EC Cloud Select Industry Group (C-SIG), 2014).
 - o Deliverables produced during the project (primary data) are retrievable in Appendix B: Business Process as a Service.
- Unstructured interview with the industrial project partner Mathema¹⁸ (primary data) - an Italian company operating in the IT field since the 1987 and expert in supporting SMEs – to identify appropriate Cloud solutions. The interview had the purpose to increase understanding on how to ask business-like questions to retrieve Cloud solutions. The results of the interview are retrievable in Appendix B: Business Process as a Service, folder B7 while the consolidated result is reported in Sub-section 4.2.6.5.
- Data about cloud services were collected from four cloud marketplaces - Ymens¹⁹, IBM²⁰, Also²¹ and UK digital marketplace²² - (primary data) contributed to create the non-functional requirements for Cloud solutions as well as real-world cloud services in the ontology (see Sub-section 4.2.6.5).
- Related scientific work (secondary data) – see Sub-section 4.2.2. This includes the analysis of 46 scientific papers (i.e. from 2009 to 2018) focusing on the description of non-functional requirements of cloud services (see Sub-section 4.2.6.5).

3.6.2 Source Triangulation in the Design Science Research

As already mentioned at the beginning of the sub-section, different data sources were considered in this research with the objective of conceiving a set of requirements to be fulfilled by the new artefact. The data sources, thereby, are the following:

- One case targeting the Patient Transferal Management application domain (Section 4.1).
- One case targeting the Business Process as a Service application domain (Section 4.2).
- Modelling experts, who were interviewed to identify the needs for domain-specific adaptations of modelling languages in the industry. Sub-section 4.4.2 describes the technique used for the semi-structure interview as well as the results. The interview had a second part from which the tentative design of the artefact was introduced. Namely, a set of operators for the agile meta-modelling was presented. The results from the second part of the interview, therefore, helped to refine the set of operators (see Sub-

¹⁸ <https://www.mathema.com/>

¹⁹ <http://www.ymens.ro/en/frontpage>

²⁰ www.bluemix.net

²¹ www.alsocloud.ch

²² <https://www.digitalmarketplace.service.gov.uk/g-cloud>

section 5.1.6.1). In turn, the interview also contributes to answer the third research question (RQ3: *How can agility be fostered when performing domain-specific adaptations of modelling languages?*). The data sources for the results of both parts of the interview can be found in Appendix C: Modelling Expert.

- Theories – relevant literature was considered to buttress the list of requirements conceived at the end of the *awareness of problem* phase (see Sub-section 4.4.5).

3.6.3 Literature Review

The literature review helps to determine the state-of-the-art field of study (Mouton, 2001). This serves to generate and refine the research ideas. Hence, the more the literature is recent and credible (i.e. retrieved from reliable sources) the better. However, according to Saunders et al. (2009) the literature review aims at not just identifying but also critically reviewing the existing literature in a particular area of interest. The critical stance aims at ensuring validation of the literature which is fundamental to support the research question(s). For these reasons the literature search is an activity that starts early in a project research and continues along with the whole project's lifespan. The interested reader can find an extensive discussion about the literature review process as well as suggestions on how to conduct a critical review in (Saunders et al. 2009).

Saunders et al. (2009) identify two precise purposes of conducting literature review. One is of adopting a *deductive approach*, where theories and ideas are identified and tested by using data. The other is of adopting an *inductive approach*, where data are explored and a body of knowledge of the subject area of interest is created.

Gall et al. (2006), list some other purposes for conducting literature review. Among others, literature review helps researchers to refine the research question(s) and objectives. It helps to discover explicit recommendation for further research. It avoids repeating work that has been done already by someone else. Finally, it helps to discover insights into research approaches, strategies and techniques that may be appropriate to address the research question(s) and objectives.

Chapter 2 presents an exhaustive literature review, organised according to the relevant topics of this research work. As mentioned in Sub-section 3.3.2, additional literature was investigated to support the different phases of the design science research cycle: awareness of problem, suggestion, development, evaluation and conclusion.

3.6.4 Interviews

According to Kahn and Cannell (1957) interviews are purposeful discussions among people. This data collection technique is well-known to provide valid and reliable data that is beneficial to address research question(s) and objectives. An interview enables the researcher to get a close contact with interviewees, creating environments for welcoming opinions and suggestions, which enrich the data collection (Shneiderman & Plaisant, 2004). There exist several types of interviews. According to Saunders et al. (2009) three categories can be distinguished, i.e. *structured interviews*, *semi-structured interviews* and *unstructured* (or *in-depth*) *interviews* and *focus groups*. Based on the work of Healey and Rawlinson (1994), each of these type can be further distinguished as either standardized or non-standardized typologies. While structured interviews are considered as being standardized and often referred to as quantitative data collection, the semi-structured, unstructured interviews and focus groups are non-standardised and referred to as qualitative data collections (King, 2004). Saunders et al.

(2009) suggest adopting the type of interview according to the research question(s), objectives, the purpose of the research and the adopted research strategy.

In this work, unstructured, semi-structure interviews and focus groups are employed. In the following, these three data collection techniques are further elaborated together with their strengths and weaknesses. At the end of each Sub-section the data collection technique is contextualised to this research work.

3.6.4.1 Unstructured and Semi-Structured Interviews

Unstructured interviews are informal, one-to-one interviews using which researchers can explore in depth the area of interest by asking open-ended questions to the participant (Robson, 2002). These do not follow any fixed protocol with predetermined list of questions like it is for the structured interviews and partially for the semi-structured interviews. However, aspects to be explored must be clear to the interviewer. This type of interview embeds a high degree of freedom, which allows collecting a rich and detailed set of data. According to Saunders et al. (2009), the interview is usually guided by the interviewee's perception. The latter are likely to lead the discussion into not previously considered areas, but which are relevant for the understanding of the particular research topic, and thus helps addressing the research question(s) and objectives.

In the semi-structure interview, the researcher has a list of questions to be covered and these may vary from interview to interview (Saunders et al., 2009). In other words, depending on the relation between the context and the research topic, some questions may be omitted. Depending on the flow of the conversation, the order of the questions may shuffle. Questions that were not foreseen beforehand may be required to explore the topic under investigation. Saunders et al. (2009) suggest using techniques of audio-recording or note taking during the conversation because the nature of the questions and the ensuing discussion.

Saunders et al. (2009) suggest conducting unstructured and semi-structured interviews via face-to-face meetings, phone calls or internet such as email, internet forums or chat rooms. Each one of them has some advantages and disadvantages, like for instance in an early stage a phone call could result to be more appealing than a face-to-face meeting due to financial, resources and time reasons. However, the latter allows building a better personal relationship between interviewer and the interviewee which (if appropriate) increases the quality of data and ensures a better response rate. Despite the rich data that can be gathered via unstructured and semi-structured interviews, the way of interacting with subjects and asking questions is crucial to determine a good quality of data (Silverman, 2007). Hence, it is up to the interviewer's skills to ensure quality in the collected data, which enables him/her responding the research questions. Saunders et al. (2009) identify three issues related to the quality of data when using non-standardized interviews, i.e. reliability, forms of bias, validity and generalizability of data. Silverman (2007) associates the reliability issue with the retrieved information being independent from the interviewer; in other words repeating the same interview with a different interviewer should result with similar data. Saunders et al. (2009) link the reliability issue with forms of bias in the sense that interview's responses' interpretation or presumptuous inclination may result in bias data, hence reducing reliability of the collected data. Also in this context validity of data is threatened since it is characterized by the way a researcher extrapolates knowledge from interviews. Saunders et al. (2009) suggest that in order for the data to have a high level of validity, questions should be clarified, meanings of answers should be investigated, and topics should be discussed from several facets. Finally, the generalizability (or transferability) issue is given by the small number of cases that the qualitative nature of the study imposes. In this case, the researcher should promulgate that his/her findings apply on a broader spectrum. This can refer to settings of a case that can be

extended to a larger scale as well as can prove that findings advance existing theoretical propositions (Marshall & Rossman, 1999).

In this work the performed interviews were the following:

- Two unstructured interviews in each of the created case. Details are reported in Sub-section 3.6.1.
- Semi-structure interviews to modelling experts. Details are reported in Sub-section 3.6.2.

3.6.4.2 Focus Groups

Focus groups are effective and well-established methods to evaluate an artefact in design science research (Stewart et al., 2007). According to Hevner and Chatterjee (2010) focus group methods are adopted on one hand to refine the artefact design (also known as exploratory focus group “*EFGs*”), and on the other hand to establish the utility of the artefact in the application field (also known as confirmatory focus group “*CFGs*”). A focus group comprises of a discussion of a topic, generally up to two hours, among 6-12 people under the direction of a moderator. As the term “focus” suggests, questions should be addressing a small number of concerns. Hevner and Chatterjee (2010) state that focus groups are utilized as a source of primary data in basic and applied behavioural disciplines such as health sciences, sociology, management, and organizational behaviour. Recently, it has become much relevant in Information Systems research (Baker & Collier, 2005) as well as in software engineering (Kontio, Lehtola, & Bragge, 2004). Krueger and Casey (2000) believe that in focus groups people interact and influence each other, therefore focus groups provide more natural settings than individual interviews. In such an environment shared understandings and individual differences of opinion are raised. As Stewart et al. (2007) suggest, there persists additional advantages to adopt focus group as an evaluation technique for design science research. Among others, focus groups are flexible as they can handle a broad spectrum of design topics and domains, they create a direct interaction with respondents, and they provide a large amount of rich data. Finally, focus groups allow building on other respondent’s comments which (according to Stewart et al. (2007)) might be missing with individual interviews. From this aspect new issues regarding the proposed artefact may arise.

There are no ordained steps to define and design focus groups. Authors like Hevner and Chatterjee (2010) suggest certain steps, however, the ultimate determination of the these steps depends only on the research intent.

In this research work, a focus group was executed to evaluate only the first DSML developed within the Patient Transferal Management application domain. For the second DSML, it was not necessary as feedback were continuously provided by the project consortia on each new version of the DSML. As mentioned in Sub-section 3.6.1, the focus group (including the design in line with the research intent) is described in Sub-section 4.1.6.4 and the data source is reported in Appendix A: Patient Transferal Management, folder A11.

3.6.5 Data Analysis

The objective of data analysis in qualitative research is to create an understanding and interpretation of the collected data, which contribute to answer the research question(s) thoroughly (Newton, 2012; Merriam, 2009; Saunders et al., 2009; Kaplan and Maxwell, 2005). Maxwell (2013) states that analysis of data commences at the beginning of a research and concludes with the research findings. According to various authors, such as Merriam (2009) and Kaplan and Maxwell (2005), qualitative data collection and analysis take place

simultaneously. It is an on-going activity that makes sense of any piece of data just as it comes in. Newton (2012) refers to this as a process of “*organization, reduction, consolidation, comparison, and reconfiguration*”. Agar (1980) describes it as follows:

“you learn something (“collect some data”), then you try to make sense out of it (“analysis”), then you go back and see if the interpretation makes sense in light of new experience (“collect more data”), then you refine your interpretation (“more analysis”), and so on. The process is dialectic, not linear”.

According to Saunders et al. (2009) the qualitative data analysis techniques assist researchers to develop theory from data, which best suits the inductive research approach. The inductive approach (see Section 3.2) sets forth by collecting data which are then interpreted to determine what subsequent themes to focus on. This process leads to the construction of theory (Glaser & Strauss, 1967). Yin, (2003) argues that difficulties arise if data are not analysed as they are collected. Although data collected with qualitative techniques are rich in meaning, it is likely to deteriorate over time unless it leaves the researchers’ heads and takes another useful form. This would lead to a distorted outcome, which according to Yin, (2003) fails the purpose of data analysis. To assist in this collection-analysis process, some authors suggest qualitative analysis techniques such as *summarising (condensation) of meanings* (Saunders et al., 2009), *coding* (Myers, 2009), and *analytical memos* (Kvale, 1996). Saunders et al. (2009) suggest developing a sort of conceptual framework with the result of each qualitative analysis. The conceptual framework can then guide the data collection follow up and analysis sessions.

Condensation and analytical memos were used in this research work as qualitative analysis techniques and are briefly described below.

3.6.5.1 Condensation

Condensation or summarising of meanings “*involves condensing the meaning of large amounts of text into fewer words*” (Saunders et al., 2009). As soon as the researcher starts summarising information, he or she is already in the process of analysing data. This process let emerge; the relevant themes from the conducted interview and the route of exploring these in the upcoming data collections. Additionally, this technique allows identifying relationships between themes, upon which theories can be developed (Kvale, 1996).

In this work, the collected data (including their meaning) from interviews and collaborations with project members were transcribed in *documents* or *Power-Point presentations* in the form of a written summary or conceptual model. Power-Point presentations were mainly used to store information from collaboration as one could quickly draw meaningful sketches. Documents created from unstructured, semi-structured and focus group interviews were all acknowledged by the subjects and sometimes even completed with additional remarks. As mentioned in Sub-section 3.6, available documents pertaining to the application domain and therefore relevant to increase understanding were also considered as data source. This qualitative analysis of data from multiple sources (i.e. triangulation) allowed acquiring a richer understanding of the phenomenon under study.

3.6.5.2 Analytical Memos

According to Kaplan and Maxwell, “an analytical memo is anything that a researcher writes in relationship to the research, other than direct field or transcription” (2005). Analytical memos can be set out informally. They can also take forms of written comments to attach on transcript(s), either notes about an idea concerning the research or a complete essay. Their purpose is to facilitate reflection, an analytical insight to a researcher and apply any manipulation on them (Strauss & Corbin, 2015).

Saunders et al. (2009) argue that memos should be written while engaging in interviews, while interviews are written and/or transcribed, while data are categorised and refined, while studying organizational documents, while writing the research project and so forth.

The memo is considered to be an important analysis technique, and as Wolcott (2001) suggests it should be produced prior to the commencement of the research. Miles and Huberman (1994) add that it is helpful to write down the date for each memo and if they are grounded in data, to associate them to the data source. When necessary memos can also be categorised and used in the coding process (Saunders et al., 2009).

In this work analytical memos are applied on the collected data and combined with the condensation technique.

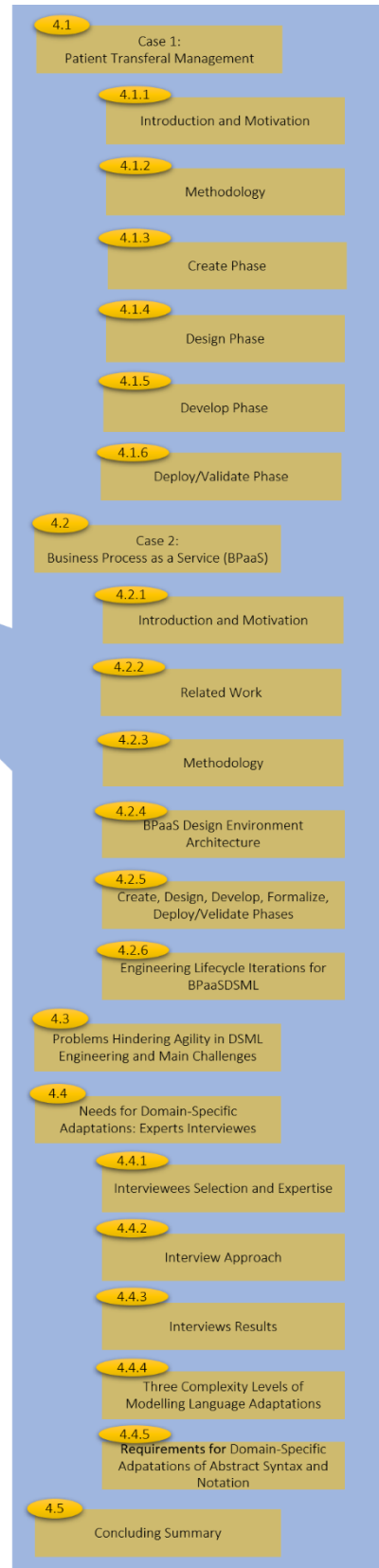
3.6.6 Ethics of the Research

All participants involved in data collection activities of this research work volunteered and have signed the consent. The Faculty Committee for Research Ethics and Integrity of the University of Pretoria has granted approval for the application with reference number EBIT/11/2018. Therefore, this research study meticulously adheres to the ethics of The University of Pretoria.

3.7 Concluding Summary

This chapter described the scientific procedure that manoeuvres this research work. The relevant aspects of this procedure was supported by the 'research onion' proposed in (Saunders et al., 2009; Saunders et al., 2019). A layer of this research onion comprises of certain stances, each of which has been buttressed by a theoretical background. The philosophical underpinning of this research work is pragmatism and the research approach is inductive. A case study research supplements the main research strategy of this work, namely design science research. The multi-method qualitative research conducted is characterized by a triangulation approach, which corroborate the research findings and is employed on both data sources and methods to amplify the reliability of data, in terms of credibility and validity. Details on employed data collections techniques and data analysis procedures were described and are tailored to the research questions and objectives.

4. PROBLEM RELEVANCE AND CASES



This chapter addresses the research questions RQ1 and RQ2:

(RQ1) What are the problems that hinder agility in domain-specific adaptations of modelling languages?

(RQ2) What are the needs for domain-specific adaptations of modelling languages?

The answers to the two research questions contribute to the first phase of the design science research (DSR) approach, which is the *Awareness of Problem* phase (see Sub-section 3.3.2).

Specifically, in order to answer RQ1, understanding of the addressed problems was increased by embracing a case study research strategy. As already clarified in Sub-section 3.3, the case study research supplements the main research strategy of this work, which is the Design Science Research (DSR). The case design is reported in Sub-section 3.3.4 and follows the guidelines of Yin (2003). Given higher benefits of multiple cases (see Sub-section 3.3.3), two case studies have been created and analysed. Both cases follow the same case design, which has the following proposition:

The creation of the DSML for each case follows the *AMME Lifecycle* methodology for three main reasons, which were identified from the investigation of the literature (see Sub-section 2.10.4). Namely, the AMME Lifecycle

- is the most advanced in incorporating agile principles.
- allows performing direct domain-specific adaptations of the meta-model.
- is widespread in the Enterprise Modelling research field, and it has already successfully followed to develop other DSMLs.

To further increase the generalisation of findings, each case addresses a different application domain:

- Patient Transferal Management (Section 4),
- Business as a Service (BPaaS) (Section 4.2).

As reported in the literature findings (see Sub-section 2.6.2), the type of knowledge representation is a distinguishing factor for DSMLs. Namely, DSMLs are used for the human-interpretable representation of knowledge (i.e. through the graphical models), but can also be interpreted by machines, through ontologies. In order to enrich findings, the Patient Transferal Management case focuses on the human interpretation of knowledge while BPaaS extends to the machine-interpretable of knowledge. Moreover, both DSMLs were implemented with ADOxx²³ Development Toolkit (Fill & Karagiannis, 2013), which allows for the customisation of meta-models (see Sub-section 2.8.2) and is widely adopted within the Enterprise Modelling research community.

As the results of the case analysis, Section 4.3 reports a consolidated list of problems and challenges to be tackled by the new agile meta-modelling approach.

Next, to answer research question RQ2, there was the need to understand the relevancy of domain-specific adaptations in practice thoroughly. Therefore, interviews were conducted with modelling experts from the industry, and results are consolidated in Section 4.4. A triangulation among (1) the findings from the interviews (2) the lessons learned from the development of the two DSMLs, and (3) literature review, led to conceive the requirements (Section 4.4.5) to address by the new agile meta-modelling approach.

²³ <https://www.adoxx.org/live/home>

4.1 Case 1: Patient Transferal Management

During the project Patient-Radar²⁴ (Reimer & Laurenzi, 2014), it turned out that the standard BPMN is not appropriate for modelling transferal management processes. Therefore, a DSML for patient transferal management (DSML4PTM) was developed (Laurenzi et al. 2017). This development gave the chance to analyse the issue from a practical perspective – in addition to the theoretical findings. DSML4PTM focuses on the human interpretation of knowledge.

This case study is structured as follows; first, the motivation to create the DSML is provided, then methodology AMME Lifecycle is shortly discussed. In remaining chapter, different phases of AMME cycle are discussed.

Each phase is reported in a dedicated sub-section. Problems hindering agility of the domain-specific adaptation are reported at the end of each phase.

4.1.1 Introduction and Motivation

Recent statistical findings released by Eurostat (2017) revealed that healthcare is still the main expenditure in developed countries. Germany is leading the list among EU members with current healthcare expenditure equivalent to 11.0% gross domestic product (GDP), while Switzerland has an even higher expenditure with 11.4%. More increase is expected according to World Bank (2014). And the governments are reacting to this situation by imposing pressure on healthcare providers, especially on hospitals to lower the cost. However despite these tough conditions, hospitals should provide a high quality of service (Lenz et al., 2012). Which is only possible if processes and activities are as optimal as possible. This is a huge challenge due to the complexity of the domain. Many structured and ad-hoc processes that involve a broad range of crucial decisions typically can take place across organizations and among actors with different expertise. One process that reflects such a complex environment is the so-called *transferal management* process, also known as transitional care or hospital discharge management or planning. Parry et al. (2008) defined transferal management as “a set of actions designed to ensure the coordination and continuity of care received by patients as the transfer between different locations or levels of care”. This set of actions, also called administrative pathways, includes medical information and excludes the treatment of the patient, which is referred to as clinical pathways (Lenz & Reichert, 2007). The Patient Radar Project (Reimer & Laurenzi, 2014) addresses the issues mentioned above by enabling intersectoral collaboration between acute hospitals and rehabilitation clinics, where (a) rehabilitative expertise is brought early into the acute somatic treatment loop, and (b) demand for rehabilitation treatment is needed as early as possible. Such collaboration takes place within the complicated settings of the transferal management domain, where many domain experts are involved, i.e. from acute hospitals, rehabilitation clinics, and health insurance for cost reimbursements.

In order to provide all the relevant concepts and decision types of the transferal management application domain, a DSML was developed, called DSML4PTM. This DSML offers graphical notations which are known to domain experts such as physicians and transferal managers. This is done so that domain experts design and adapt models.

²⁴<https://www.fhsg.ch/en/projects/project/patient-radar-226/>

4.1.2 Methodology

As depicted in Figure 33, the AMME Lifecycle (Karagiannis, 2015; 2018) (see also Sub-section 2.10.4) was instantiated to create DSML4PTM. Each phase was considered except the “formalize” phase, as it was not required from the application domain. Two types of requirements were considered to create the DSML; first, the theory-based requirements (TBRs) and second, the application domain-based requirements (ADBRs). The requirements were considered in the initial phase “create”, see TBRs and ADBRs on the bottom-left-hand side of Figure 33. Theory-based requirements refer to the requirements elicited from the knowledge base of scientific foundations and experience. Mainly, existing DSMLs in the healthcare that are built through domain-specific adaptations are considered to derive the TBRs (see Section 4.1.3). This approach provides the theoretical foundation for the design of DSML4PTM. Workshops with modelling and domain experts were conducted to ensure relevancy of the DSML4PTM in the application domain. Further on, hospital-related documentation and health care standards were also considered. All this knowledge was used to derive the application domain-based requirements.

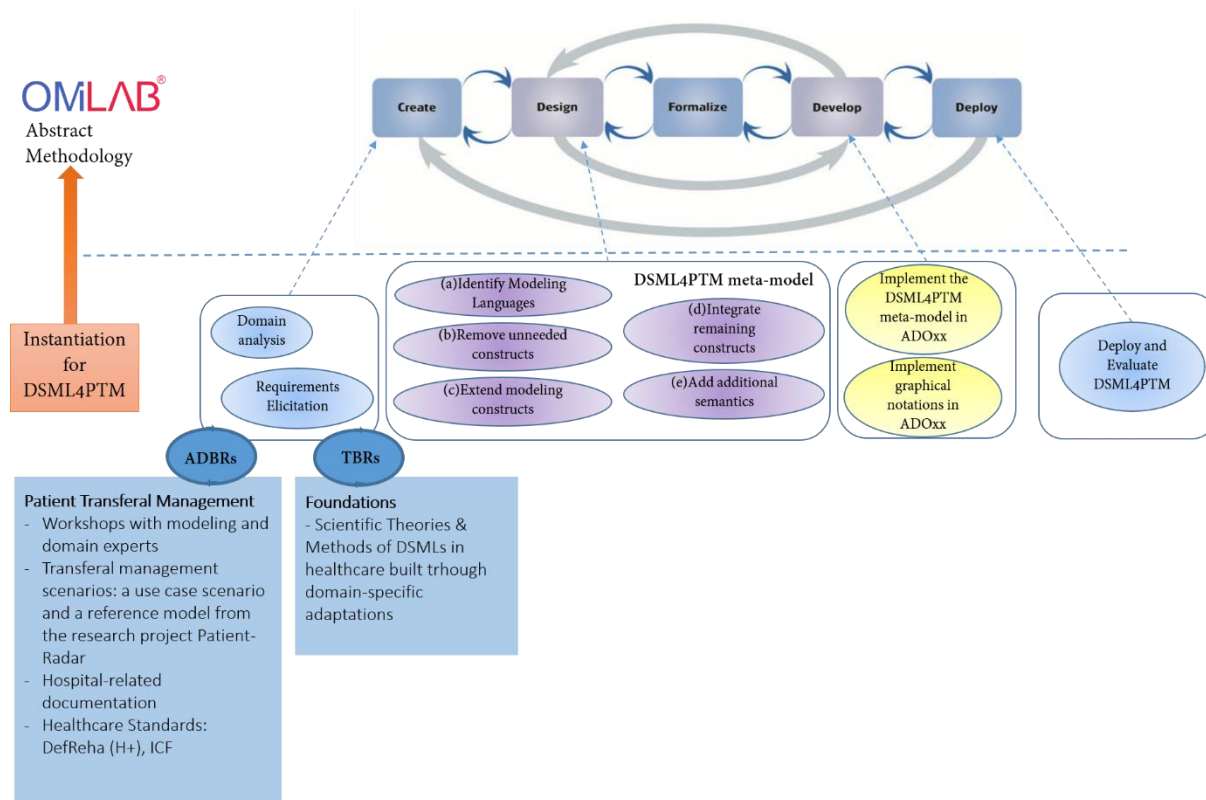


Figure 33. Instantiation of the AMME Lifecycle (Karagiannis, 2015; Karagiannis, 2018) for creating DSML4PTM

Each instantiated phase of the AMME Lifecycle is briefly and elaborated below.

- **Create phase:** This phase aims to increase understanding of the application domain and relevant scientific theories (Sub-section 4.1.3). On one hand, existing domain-specific adaptation approaches that studied DSMLs in healthcare are investigated. From this investigation, theory-based requirements (TBRs) are derived. Further on, understanding is deepened by analysing the application domain. From this activity, the domain-based requirements (ADBRs) are derived.

- *Design phase*: This phase deals with the conceptualization of the meta-model (Sub-section 4.1.4). For this, both requirements TBRs and ADBRs elicited in the previous phase are addressed
- *Development phase*. This phase deals with the development of the DSML4PTM (Sub-section 4.1.5). The meta-model conceptualised in the previous phase is implemented in the ADOxx Development Toolkit, i.e. meta-modelling tool. Each new modelling construct is implemented with a new graphical notation.
- *Deploy/Validate phase*. The deploy/validate phase deals with the deployment of the modelling language in a modelling tool as well as evaluation of the modelling language and models (Sub-section 4.1.6). That is first, the developed DSML4PTM is deployed from the meta-modelling tool into an ADOxx-based modelling tool. The modelling language was then used in the modelling tool for evaluation purposes. The evaluation of DSML4PTM was performed (1) Concerning the elicited requirements; (2) by modelling both the reference model and application scenario created in the awareness of the problem (3) by conducting a focus group with modelling and domain experts. In the focus group, models had to be adjusted by both domain and modelling experts. This activity was used to evaluate the perceived usefulness and cognitive effort of the created DSML4PTM. From this phase feedback generated loop back to the first phase.

As shown in the AMME Lifecycle in the intertwining of phases was considered to inject agility in the development of the DSML.

4.1.3 Create Phase

This section describes the instantiation of the Create phase of the AMME Lifecycle. First, literature is investigated to derive theory-based requirements. After that, the domain analysis is performed from which application domain-requirements are elicited.

4.1.3.1 Theory-based Requirements (TBRs)

The DSML4PTM focuses on administrative pathways of the patient transferal management. At the time of conducting the project Patient-Radar there was no DSML with the purpose of modelling administrative pathways at general level and neither specifically for the patient transferal management.

However, several DSMLs exist in healthcare. It is increasingly common to support experts such as physicians (or other hospital personnel) with domain-specific models that conceptualise an underlying reality. Most recent works mainly focus on modelling clinical pathways (CPs), which include treatment activities, e.g. (Burwitz et al. 2013; Heß et al. 2015; Herzberg et al. 2015; Braun, Schlieter, et al. 2015; Braun et al. 2016).

Other DSMLs in healthcare were developed to model:

- Intraoperative surgical workflows in operating rooms (Neumann et al., 2016);
- Communication aspects within hospitals for hospital personnel (Wu et al., 2012);
- Business process knowledge of healthcare workers (Jun et al., 2009);
- Patient's treatment protocols and guidelines (Mathe et al., 2009);
- Electronic Patient Care Records (ePCR) (Shenvi et al., 2007).

Relevant DSMLs for this work were selected. The relevance is based on the following criteria:

- The healthcare application domain of the DSMLs;
- The creation of a DSML through adaptation of the meta-model;
- The successful development and or use of the DSML either in research or in industry.

Each selected work was analysed with respect to the following three questions:

- What are the needs for the development of a DSML?
- What are the procedures for designing a DSML?
- How does the resulting DSML look like?

The analysis increased awareness of *what needs* lead to the creation of a DSML and *how* DSMLs are commonly developed in healthcare. The gained knowledge provides the theoretical foundation underpinning the design of our DSML.

The selected DSMLs are elaborated in coming Sub-sections. The sub-section ends with the list of relevant notions from the analysed DSMLs. These notions set the basis for the theory-based requirements (TBRs).

4.1.3.1.1 CP-Mod

Burwitz et al. (2013) present a dedicated DSML to model clinical pathways. The need for the domain-specific adaptation was due to the inability of existing modelling languages to model all the relevant aspects of clinical pathways. The relevant aspects were grouped into three main categories: (a) evidence-based medicine and decision support, (b) the classification of different treatment alternatives, (c) time events and waiting periods.

The engineering approach follows the following steps:

1. Elicitation of requirements;
2. Comparison of requirements with respect to concepts of existing modelling languages;
3. Aspects of existing modelling languages that fulfil the formulated requirements are considered as baseline;
4. Direct adaptation of the meta-model to accommodate the new requirements (i.e. domain-specific adaptations);
5. Abstract syntax and concrete syntax of the DSML are developed in the CASE-Tool;
6. The evaluation of the DSML is done by:
 - a. showing that all the formulated requirements are fulfilled,
 - b. demonstrating the modelling of the wisdom tooth treatment scenario.

Burwitz et al. (2013) elicited the following four requirements:

- “*Requirement R1*. A language for modelling clinical pathways should provide the basic concepts of the medical business process modelling (patient state, treatment step, decision, process flow) and the ability to integrate information objects and responsibilities.”
- “*Requirement R2*. A language for modelling clinical pathways should provide concepts for describing indefinite order relations as well as compulsory parallel relations between treatment steps and iterating treatment steps.”
- “*Requirement R3*. A language for modelling clinical pathways should provide concepts for describing evidence-class of any recommendation and linking the

source of evidence. Additionally, a concept to describe evidence-based decision is required”.

- “*Requirement R4*. A language for modelling clinical pathways should provide concepts to describe temporal dependencies and explicit time events.”

Figure 34 shows the table created in Burwitz et al. (2013) that evaluates whether a requirement is met, partially met or not met with a modelling approach. For example, see aspects of the *requirement R3*, which were mainly not met.

Requirement		R1				R2			R3			R4
		Clinical state	Treatment step	Decision	Process flow	Resources & responsibilities	Variable flow	Parallel flow	Iteration	Evidence indicator	Evidence-based decision	Time event
MIS	BPMN	●	●	●	●	●	—	○	○	—	—	○
	Activity Chart	●	●	●	●	●	—	○	○	—	—	○
	Clinical Algorithm	●	●	●	●	—	—	○	○	—	—	—
IT	GLIF	●	●	●	●	○	○	○	○	○	—	○
	Guide	●	●	●	●	○	○	○	○	—	—	—
	PROforma	—	●	●	●	○	—	○	○	○	—	○
	Prodigy	●	●	●	●	—	—	—	—	○	—	○
	EON	●	●	●	●	○	—	—	○	○	—	○

● Requirement met ○ Requirement partly met — Requirement not met

Figure 34. Compliance of requirements of the modelling approaches (Burwitz et al., 2013)

Some basic concepts were taken from existing modelling languages, e.g. Clinical Algorithm notation and BPMN (lane concepts to define responsibility roles). These concepts were adapted, whereas other concepts were built from scratch to create a new DSML called CP-Mod. As an example, Figure 35 shows a meta-model excerpt from CP-Mod that fulfils *requirement R3*.

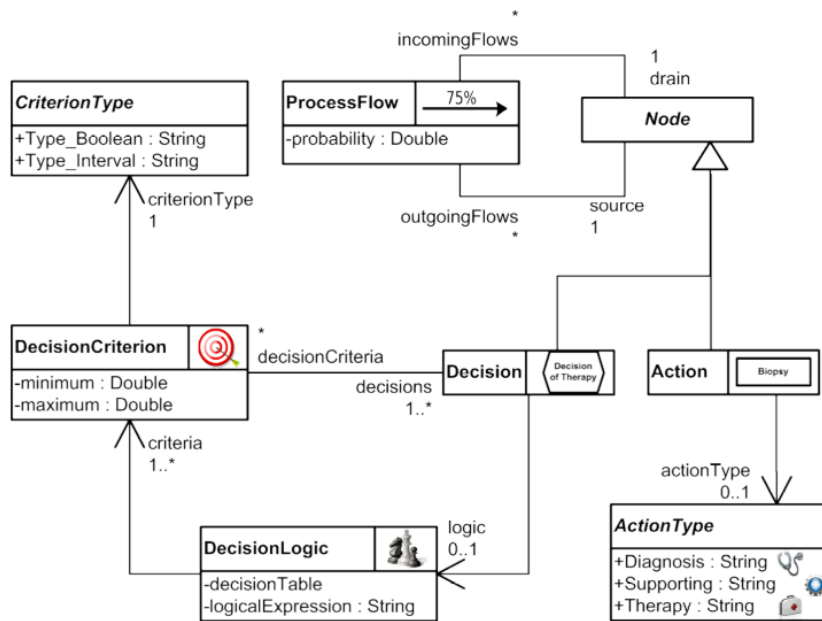


Figure 35 CP-Mod meta model excerpt – Evidence-based Decision and categorised Action (Burwitz et al., 2013)

Figure 36 shows a model developed with CP-Mod.

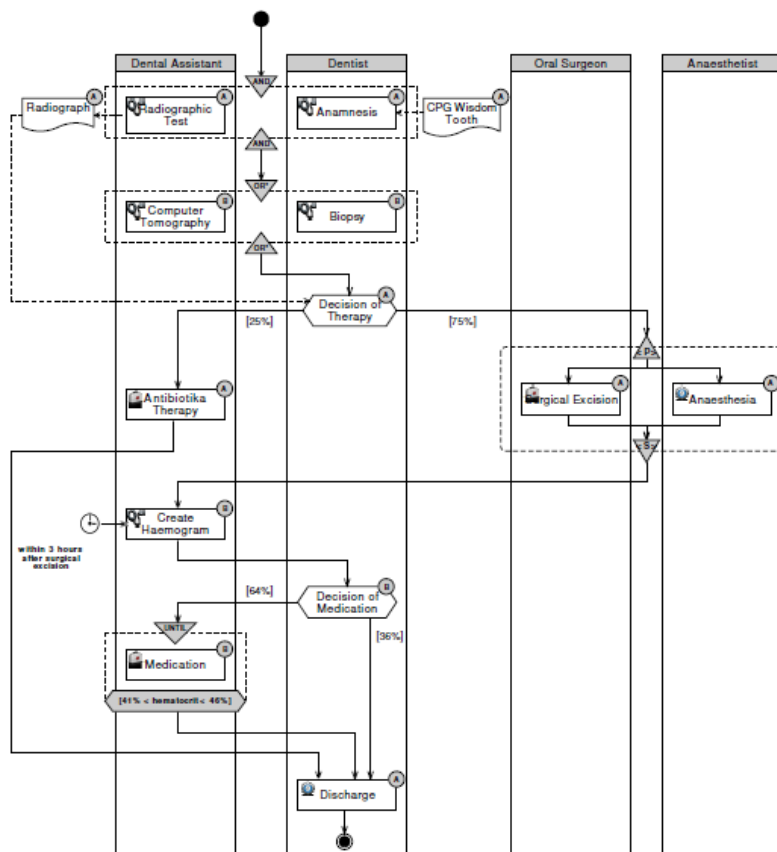


Figure 36. CP model using CP-mod (Burwitz et al., 2013)

4.1.3.1.2 DSML4CP

Similarly, Heß et al. (2015) evaluated several modelling approaches with respect to the gathered requirements in the CPs domain. None of the existing approaches fully satisfied the requirements. Although CP-Mod was among the most appropriate candidates, it was excluded due to its inability to represent relevant aspects of hospitals as organisations. The latter were fulfilled by the MEMO OrgML, which is the main motivation for choosing it as a baseline for domain-specific adaptations. In result the authors developed DSML4CPs, with a special focus on oncology.

The design of DSML4CP was underpinned by the method proposed in (Frank 2013a; Frank 2010) (see Figure 23). The authors went through the following seven steps:

1. Clarification of scope and purpose;
2. Analysis of general requirements;
3. Derivation of specific requirements using a set of scenarios;
4. Specification of abstract syntax;
5. Provision of a graphical notation;
6. Implementation of both abstract syntax and graphical notations in the MEMO's modelling tool MEMOCenterNG (Gulden & Frank, 2010);
7. Evaluation with respect to the application of the DSML4CP.

Like in the previous DSML, requirements were elicited and then compared to the possible modelling languages to adopt. Figure 37 shows the comparison and for each requirement Heß et al. (2015) indicated whether it is fulfilled, partially fulfilled or not fulfilled at all by the modelling languages.

Approach	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10
Information Systems – enterprise modelling approaches										
BPMN (OMG, 2011; Scheuerlein et al., 2012)	○	◐	○	●	○	◐	○	○	◐	○
EPC (Perrevort, 2003)	○	◐	○	●	○	◐	○	○	◐	○
MEMO OrgML (Frank, 2011a) with extensions (Heise et al., 2010; Heß, Schlieter, and Träger, 2012)	◐	●	◐	●	◐	○	○	○	○	○
Information Systems – process modelling approaches										
Aspect-oriented process modelling (Meiler, 2005)	○	●	◐	●	○	○	●	●	◐	○
CP-Mod (Burwitz, Schlieter, and Esswein, 2013)	◐	●	◐	●	◐	○	○	○	○	○
Perspective-oriented process modelling (Färber, Jablonski, and Schneider, 2007)	○	●	◐	●	○	○	○	●	◐	○
Information Systems – formal language-based approaches										
CONFlexFlow (Yao and Kumar, 2013)	◐	◐	○	●	○	○	○	○	●	○
Medicine and medical informatics field										
Clinical Algorithm (Society for Medical Decision Making, 1992)	◐	◐	○	●	○	○	○	○	○	○
GLIF (V3) (Boxwala et al., 2004)	◐	●	○	●	○	○	○	○	○	○
PROforma (Sutton, Taylor, and Earle, 2006)	◐	●	○	●	○	○	○	○	○	○
Sage (Tu et al., 2007)	◐	●	○	●	◐	○	○	○	◐	○

Legend: ○ = not fulfilled; ◐ = partly fulfilled; ● = fulfilled

Figure 37 Compliance of requirements of the modelling approaches (Heß et al., 2015)

Figure 38 shows an excerpt of the meta-model of DSML4CP, which extends MEMO OrgML. For example, the top-left of the figure depicts the three concepts identified in the requirement R3, i.e. Evidence Classification System, Level of Evidence and Grade of Recommendation (see classes with headers in grey colour).

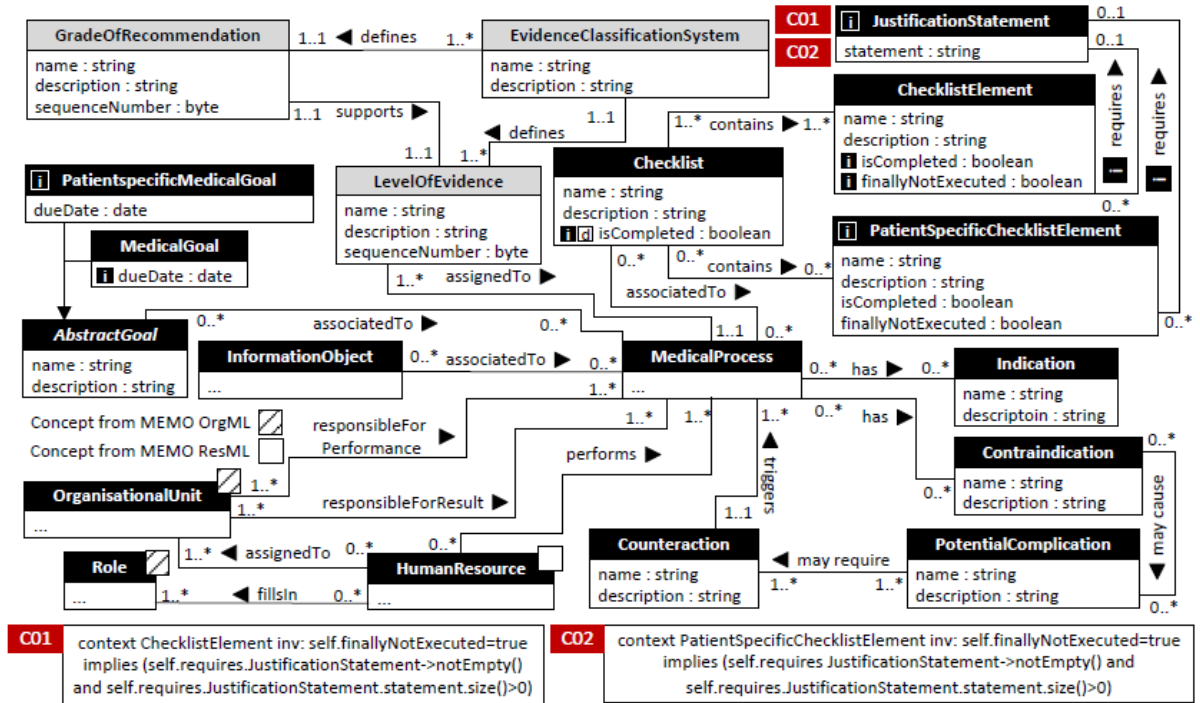


Figure 38. Meta-model excerpt of DSML4CP (Heß et al., 2015)

Finally, Figure 39 depicts a model built with DSML4CP.

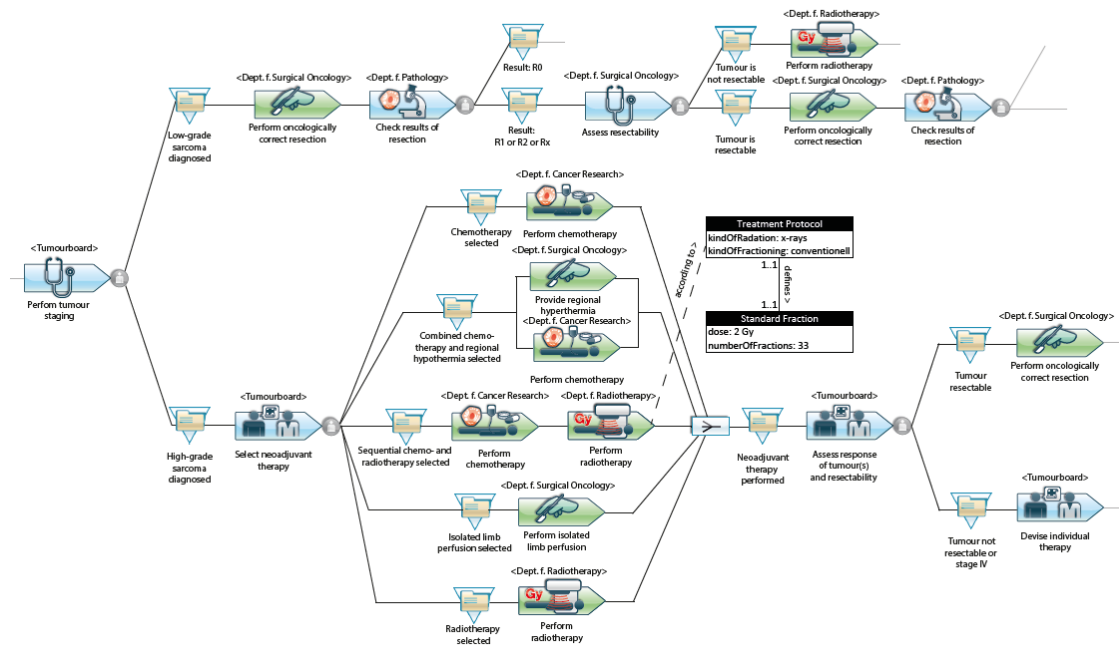


Figure 39. Excerpt from the Soft Tissue Sarcoma clinical pathway process model (Heß et al., 2015)

4.1.3.1.3 BPMN4CP

Braun et al. (2015) also faced the challenge to model Clinical Pathways (CPs). However, their focus was more into modelling the sequence flow of CPs. Therefore, the authors decided to perform domain-specific adaptations over BPMN. The authors also motivate the choice of adapting a modelling language like BPMN by stressing the benefits from the combination between a standard language and domain-specific aspects, i.e. widely established semantics and higher expressivity, respectively. As an additional argument, the authors claim that the model usage of dedicated DSML is limited from the tool chain in the DSML-world. That is, a dedicated DSML always requires its dedicated modelling tool to be used. In contrast, DSML that originate from modelling standard are better supported by existing tools and by capabilities for the model interchange. Therefore, in contrast to the previous two works, which tend to increase method pluralism (Becker, 2014; Loos et al., 2013). Braun et al. (2015) suggest to focus on (one or more) modelling standards, from which adaptations should be performed.

The development of the DSML followed the DSR methodology of Vaishnavi and Kuechler (2004). Additionally, to ensure standard-conform BPMN extensions the authors rely on the model-transformation based procedure proposed in Stroppi et al. (2011). However, the procedure lacks a deeper domain analysis and design preparation, which (according to Braun et al. (2015)) is key for an appropriate language extension. Its absence raises the risk of redundancy in the extension and can cause missing extension opportunities. Therefore, the authors adapted the Stroppi et al.'s procedure (2011). Figure 40 shows the adapted procedure, and consists of the following seven Steps:

1. Domain requirements.
2. Domain analysis and equivalence check. In the domain analysis, the authors develop a lightweight ontology to get a proper understanding of the domain. The equivalence

check compares the semantics of the needed concepts with the semantic of the concepts from the existing modelling language. In case of equivalence, there is no need to extend a concept; in case of no semantic equivalence, the new aspects need to be inserted; in case of conditional equivalence there is no apparent semantic matching. Thus, the case could fall either into equivalence or no equivalence. An extensive list of equivalence check and derivation of concepts can be found in (Braun et al., 2015).

3. BPMN concepts are extended according to the results of Step 2.
4. CMDE (Conceptual Domain Model of the Extension) model is created in UML class diagram.
5. Transforming the CDME model into an extension model (BPMN+X model). This activity creates the extended BPMN meta-model for BPMN4CP.
6. Creation of the concrete syntax for the extended concepts.
7. Evaluation of the new modelling language by demonstrating its applicability on a use case.

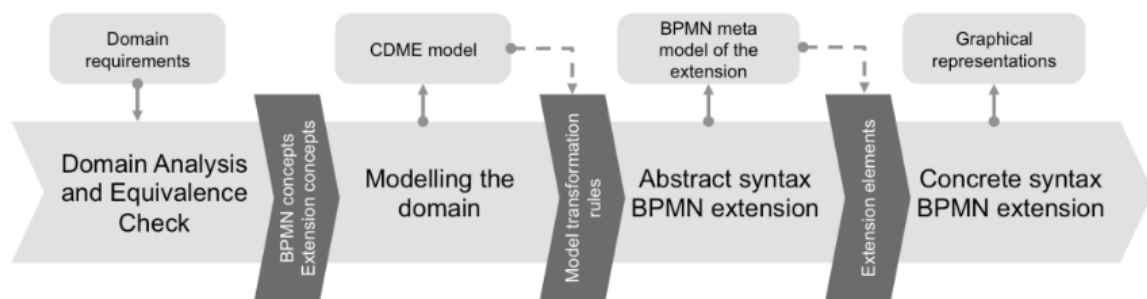


Figure 40. Procedure for BPMN extensions (Braun et al., 2015)

Eventually, BPMN4CP fulfilled the same requirements elicited for CP-Mod (Burwitz et al., 2013). Figure 41 shows an excerpt of the BPMN4CP meta-model. The classes with the tag <<Extension Concept>> (e.g. see Diagnosis Task, Therapy Task, Supporting Task) depict the new modelling elements, which extend the BPMN classes Data Object, Task and Parallel Gateway.

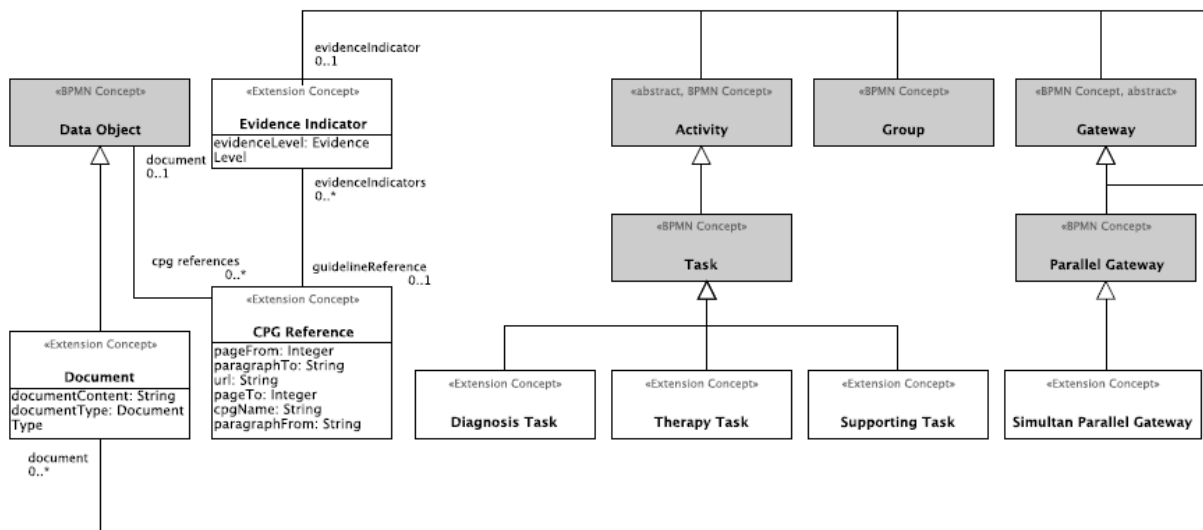


Figure 41. An excerpt of the BPMN4CP meta-model

Figure 42 depicts a concrete model built with BPMN4CP. The model depicts a wisdom tooth treatment process.

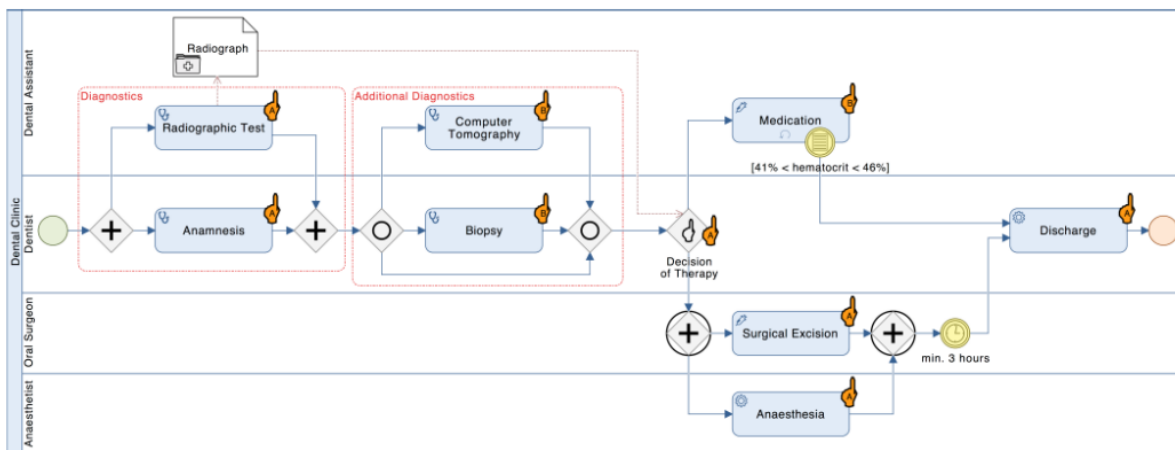


Figure 42. A wisdom tooth treatment process model built with BPMN4CP (Braun et al., 2015)

Due to additional requirements from the application domain, BPMN4CP was subject to further domain-specific adaptations (Braun et al., 2016). The new version is called BPMN4CP 2.0 and integrates the need to model resources, documents, objectives and quality indicators. Moreover, the new version of the DSML foresees the multi-perspective modelling and separation of concerns. Thus, it allows modelling separate and interlinked diagrams for resources and documents so that different stakeholder-specific perspectives can be addressed. According to Braun et al. (2016), enabling the selection of aspects for particular model users avoids the cognitive overload and misinterpretations. Further, Braun et al. (2016) stressed on necessity of transparent and replicable procedures to ensure the well-definition of BPMN extensions. Braun et al. (2016) strive to address such a need by extending the procedure depicted in Figure 40. The new procedure for the BPMN extension is shown in Figure 43 and consists of three main categories: *Domain Analysis*, *Extension Preparation* and *Extension Meta-Model*. The new steps are concrete syntax and interchange specification. To define the

concrete syntax, Braun et al. (2016) use the Diagram Definition standard from OMG. For the interchange format of the language extension, they specify the extensions with the XMI model interchange specifications and the Diagram Interchange (DI) definition.

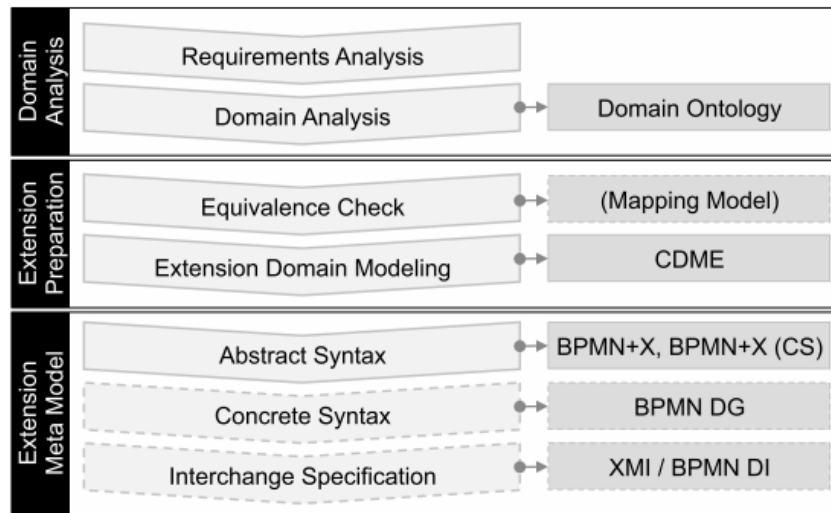


Figure 43. Integrated BPMN extension method (Braun et al., 2016)

Figure 44 outlines the abstract syntax, where the classes <<Extension Concept>> depict the new extensions for resources and documents.

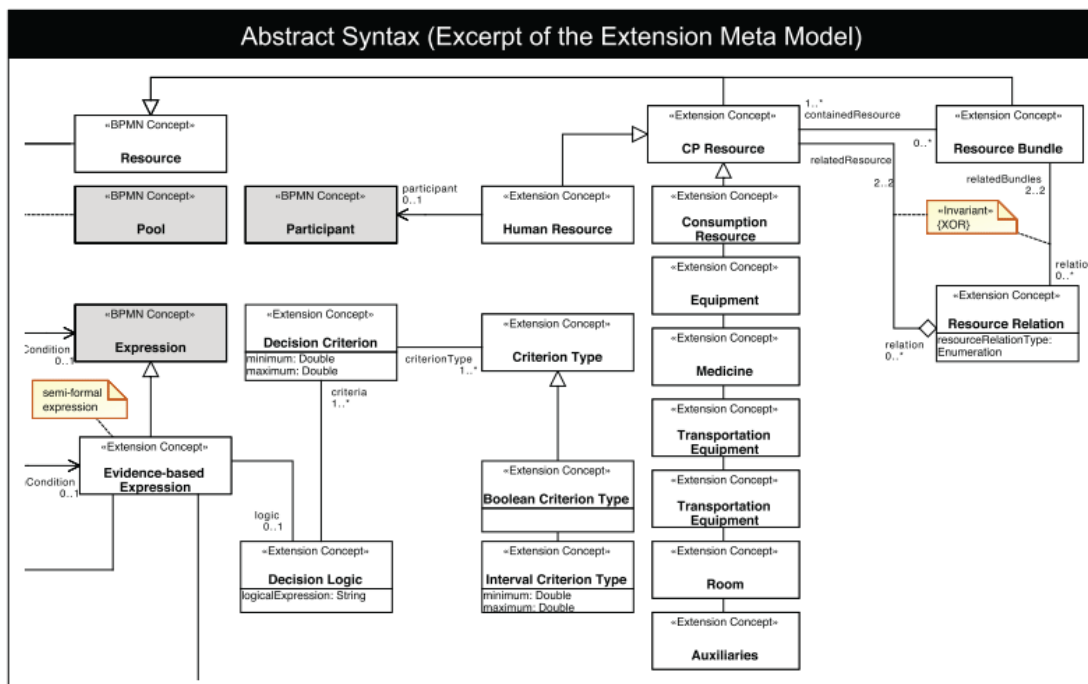


Figure 44. Abstract syntax of BPMN4CP 2.0 (Braun et al., 2016)

BPMN4CP 2.0 was validated by applying it to a stroke case. Specifically, Braun et al. (2016) modelled a treatment process for stroke patients that is used in the Stroke Network of Eastern Saxony. Figure 45 shows the stroke case modelled with BPMN4CP 2.0. Concepts from the main process model are linked to new diagrams with the dedicated views (see red dashed arrows that lead to the resource diagram and the document diagram in Figure 45). Additional

language extensions allow modelling the involved participants, time-constraints and quality indicators. The DSML was implemented in a telemedical system.

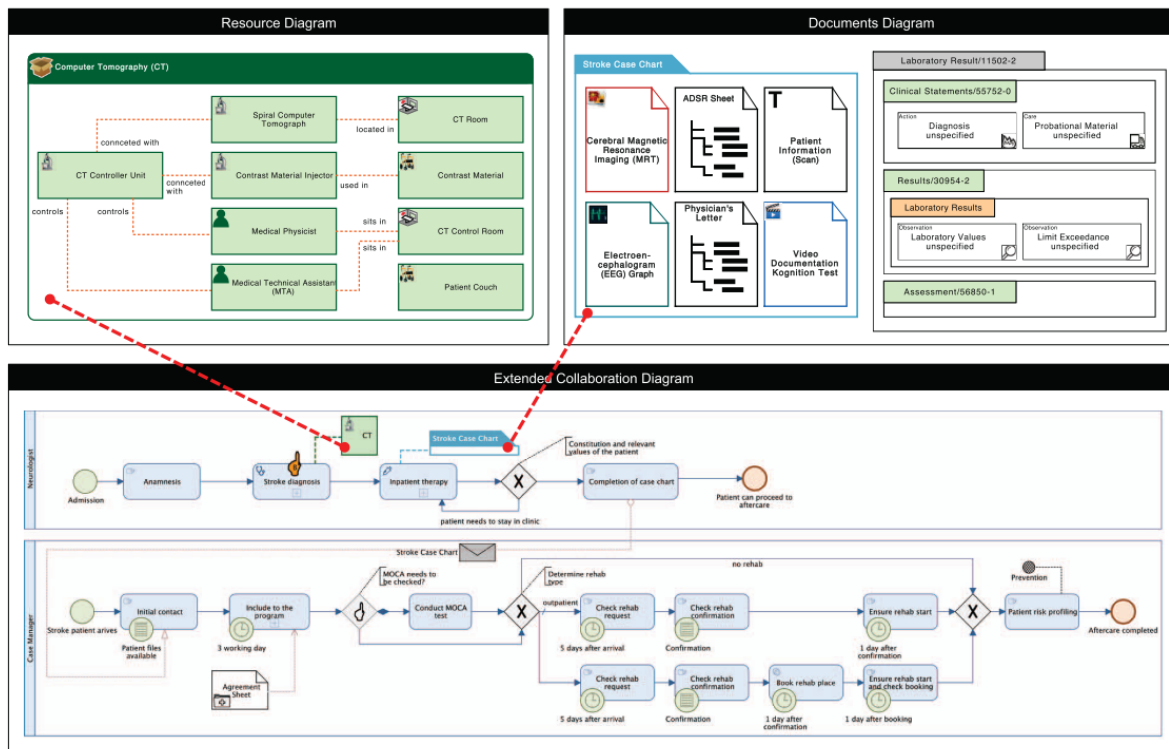


Figure 45. Stroke scenario modelled with BPMN4CP 2.0 (Braun et al., 2016)

Further on, Braun et al. (2016) suggest investigating on the adoption at the Case Management Model and Notation CMMN (OMG, 2016a). The authors consider CMMN as a suitable modelling standard to address the high degree of deviation of medical processes during run-time.

4.1.3.1.4 BPMN-SIX

Neumann et al. (2016) had to model intraoperative surgical processes and workflow in integrated operating rooms (OR). In contrast to Scheer and Nüttgens (2000), who used Event-driven Process Chain (EPC) to model OR, Neumann et al. (2016) preferred to adopt BPMN for the following reasons:

- BPMN's suitability to model surgical procedure;
- BPMN's process automation capabilities;
- BPMN's widespread distribution in academia and industry;
- BPMN's large tool support;
- BPMN 2.0 is declared ISO/IEC standard (19510:2013) for modelling and executing business processes.

Due to the general aspects of BPMN, the authors extended the abstract and concrete syntax of BPMN. The extension allows modelling domain-specific aspects of surgical workflow. Specifically, the domain-specific aspects consist of intraoperative processes, surgical activities,

anatomical structures, medical devices and clinical IT systems. The resulting DSML is called BPMN Surgical Intervention eXtension (BPMNsix).

Neumann et al. (2016) followed the same procedure proposed in Braun et al. (2015). Namely:

1. *Requirements analysis*. In order to get a better understanding of the domain, the authors developed a lightweight ontology containing all the relevant concepts. This was done to get an idea about how concepts are related with each other.
2. *Modelling languages analysis* with respect to their applicability to the modelling domain and the intended use case.
3. *Equivalence check* for the identification of BPMN extension elements (see point two of previous section).
4. *Extension of BPMN concepts* according to the results of Step 3.
5. *Development and implementation of the CMDE*. The meta-model was designed in UML class diagrams (same as point 4 in the procedure of Braun et al. (2015)).
6. *Implementation of the concrete syntax*, which includes the definition of graphical notations for the new modelling constructs.

Figure 46 shows an excerpt of the meta-model of BPMNsix. As in the previous section, classes in grey are extended by the new concepts. The latter are shown in white colour.

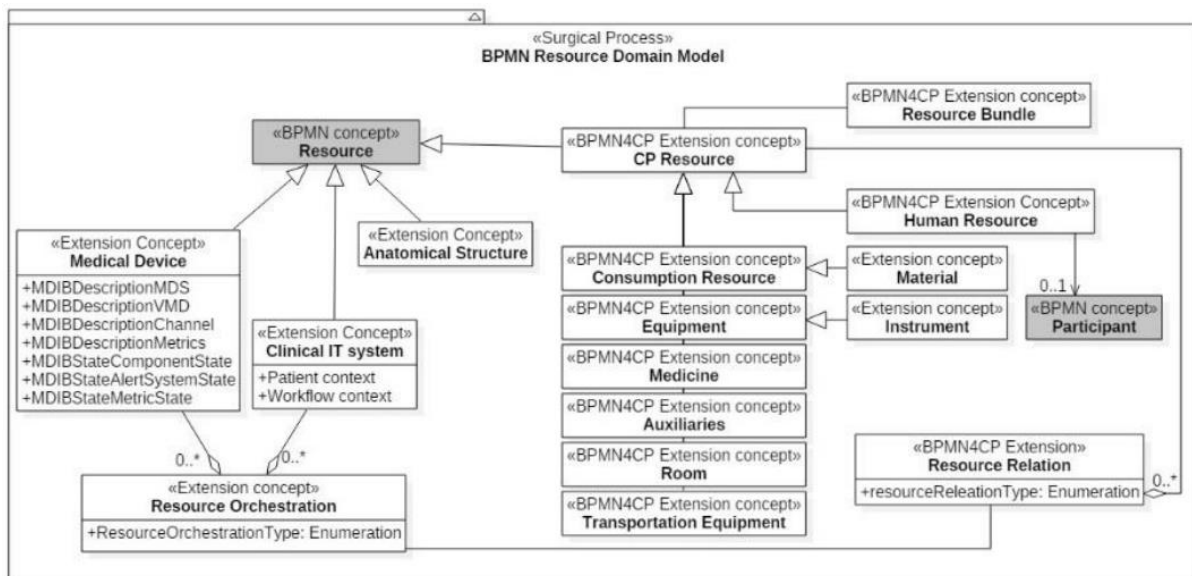


Figure 46. An excerpt of the meta-model of BPMNsix (Braun et al., 2015)

Figure 47 shows the graphical notations developed for the new modelling constructs of BPMNsix.

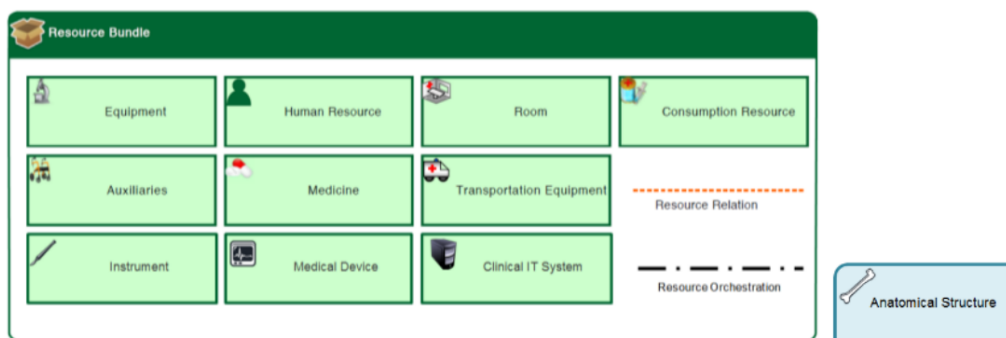


Figure 47. An excerpt of the concrete syntax for BPMNsix

Further on, Neumann et al. (2016) suggest the future research towards the adoption of the standard Decision Model and Notation (DMN) (OMG, 2016b). According to Neumann et al. (2016), it would be a valid approach to integrate DMN in to the DSML to model situation-aware surgical assistance. The DMN would be used to model rule-sets and combine them with process models and medical device models.

4.1.3.1.5 Theoretical Findings and Derivation of Theory-based Requirements (TBRs)

Four DSMLs have been presented, which fall within a healthcare application domain. Among other reviewed DSMLs, the four DSMLs were selected as they provide relevant theoretical findings for engineering our DSML. The findings are below turned into theory-based requirements for DSML4PTM. Theory-based requirements are presented in three distinct Sub-sections. Each Sub-section corresponds to one of the three questions with which the DSMLs were analysed (see also Section 4.1.3). Namely:

- Finding 1: Reasons to develop a DSML;
- Finding 2: Procedures to design a DSML;
- Finding 3: Look and feel of a DSML.

Finding 1: Reasons to Develop a DSML

The common reason for developing DSMLs in a healthcare application domain is to enable domain experts to design, adapt and use models. A DSML can help in managing following two drawbacks.

- The insufficient expressivity of existing modelling languages to model specific realities.
- Graphical notations are not cognitive adequate for domain experts (e.g. physicians or other hospital personnel).

The goal of DSML models is to support decision-making of domain experts. For this purpose, DSML models make explicit the relevant aspects (chain of activities, documents and decisions) of the addressed domain. DSML models are also integrated with run-time environments. The integration allows feeding models with aggregated data to show real-time values, e.g. to report real-time patients' conditions. Most of the above listed DSMLs contain status for certain indicators, which change according to the aggregated value of data. In alternative, run-time data are aggregated and displayed in dashboards (Mathe et al., 2009).

Within this context, a domain-expert is put in the conditions not only to use DSML models but also to support their design and to adapt them as well. Hence, the first theory-based requirement is derived as follows:

Theory-based Requirements 1 (TBR1): A DSML should enable domain-experts to design, adapt and use models.

The common reason to consider modelling standards when developing a DSML is the reusability of well-established semantics and concepts. For DSMLs where models should mainly depict a sequence flow (e.g. clinical pathways), the BPMN is rather adopted as a backbone of the DSML. BPMN (OMG, 2011) is widely used in both academy and industry, it is supported by many well-known and successful modelling tools (e.g. Bizagi²⁵, Camunda²⁶, Signavio²⁷, Trisotech²⁸). BPMN 2.0 is declared ISO/IEC standard (19510:2013) for modelling and executing business processes.

An additional lesson learned from the analysis of the DSMLs, is about the suggestion to consider not only BPMN but also other modelling standards. Braun et al. (2016) and Neumann et al. (2016) strongly suggested to investigate the use of CMMN (OMG, 2016a) and DMN (OMG, 2016b). Both standards are considered more appropriate to model cases (i.e. activities for which a predefined flow cannot be ensured beforehand) and decisions, respectively.

The adoption and integration of CMMN and DMN with BPMN have been successfully shown in the Knowledge Work Designer (KWD) modelling method (Hinkelmann, Pierfranceschi, & Laurenzi, 2016). Although the KWD does not target the healthcare domain its principles can be applied in any application domain where processes, decisions and cases need to be modelled. The KWD includes the combination of modelling languages for both *structured* and *non-structured process logic* (i.e. which activities take place – including prescribed flow and more ad hoc activities) and *business logic* (i.e. how to make decisions – including structured and unstructured logic). The differentiation of procedural from declarative information also recommended in (von Halle & Goldberg, 2010). Moreover, the KWD also integrates business process and case modelling. In the latter, BPMN and CMMN are deeply integrated resulting in a new modelling language BPCMN. With respect to modelling business logic, the modelling language spans, on one side, DMN to model structured decision logic. On the other side, the language spans the Document and Knowledge modelling language (adopted also in (De Angelis et al., 2016)) to model unstructured decision logic such as guidelines, checklist, sample outputs, or templates.

Given the additional theoretical foundation, theory-based requirements 2 and 3 are formulated as follows:

Theory-based Requirements 2 (TBR2): Application domain-based requirements should be distinguished between procedural and declarative information type as well as between structured and unstructured form. If needed, other categories should be created, e.g. documentation and organisational management or data representation.

Theory-based Requirements 3 (TBR3): The choice of the modelling languages to adapt and integrate should be driven by requirements categorization (done after fulfilling TBR2).

²⁵ <https://www.bizagi.com/en>

²⁶ <https://camunda.com/de/>

²⁷ <https://www.signavio.com/>

²⁸ <https://www.trisotech.com/>

Finding 2: Procedures to design a DSML

An additional finding that arose by analysing the above-mentioned DSMLs, is about the procedures for designing DSMLs. Such procedures supplement the Design Science Research methodology, which is, otherwise, considered too generic to develop a DSML. Hence, a design procedure is considered beneficial for the rigorous definition of the meta-model. A procedure can include steps for selecting an appropriate modelling language as well as for determining the concepts to extend. The related theory-based requirements are formulated as follows:

Theory-based Requirements 4 (TBR4): The design of a DSML through domain-specific adaptations should be underpinned by a design procedure.

A further finding concerns the domain analysis. Namely, it appears beneficial the creation of a lightweight ontology for the appropriate understanding of the domain. The requirement is, therefore, formulated as follows:

Theory-based Requirements 5 (TBR5): During the domain analysis a lightweight ontology should be developed with the relevant concepts and relations of the domain.

Finding 3: Look and feel

According to Frank (2014b), an important aspect when visualizing enterprise models is the multi-perspective and separation of concerns. Frank stresses the importance of the term *cognitive perspective*, which “represents a specific professional background that corresponds to cognitive dispositions, technical languages, specific goals and capabilities of prospective users. Hence, it is not an implicit feature of an enterprise model, but characterizes its intended purpose - to satisfy prospective users’ perspectives.” (Frank, 2014b).

Braun et al. (2016), for instance, provides separate diagrams for different stakeholder-specific perspectives. By building single diagrams while having their concepts linked to other diagrams, reduces the complexity of modelling and avoids cognitive overload.

The separation of concerns manifest itself in the Knowledge Work Designer (Hinkelmann et al., 2016), as for each type of information there is a dedicated view or diagram, i.e. the structured and unstructured procedural information type is shown in a business process diagram, the structured declarative information type is shown in decision tables, and the unstructured declarative is managed in a document view. Given the gained theoretical knowledge, the theory-based requirement is formulated as follows:

Theory-based Requirements 6 (TBR6): The visualization of a DSML should foresee the multi-perspective and separation of concerns.

DSMLs that mainly focus on representing a prescribed flow of activities consider BPMN as a backbone. That is, it reflects the main view or diagram, which may lead to further secondary diagrams. For instance, Figure 45 shows concepts from the process model lead to new diagrams (red arrows). The theory-based requirement is formulated accordingly:

Theory-based Requirements 7 (TBR7): For application domains where the main focus is the representation of prescribed flow, BPMN should be considered as the main view.

4.1.3.2 Application Domain-based Requirements (ADBRs)

In this sub-section application domain-based requirements are elicited by conducting an in-depth domain analysis. Figure 48 shows the focus of this section: domain analysis to derive application domain-based requirements (ADBRs).

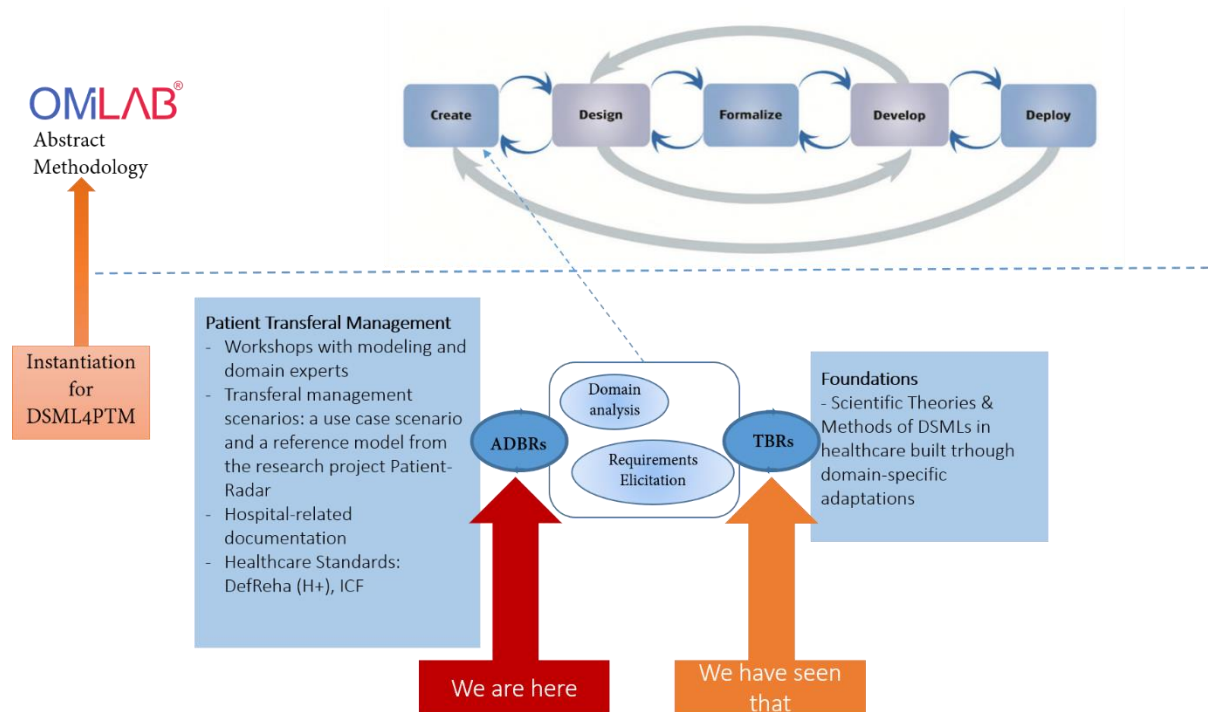


Figure 48. Instantiation of the Create phase for DSML4PTM: In depth domain analysis

This phase mainly relied on the early stage activities conducted in the Swiss research project Patient-Radar, where I was actively involved with the role of language engineer and modeller (see roles in Section 2.7). The aim of this phase was to increase understanding of the domain. Thus, several workshops were conducted with project partners. The following documents were considered as *sources* of requirements:

1. Information Model document (Sub-section 4.1.3.2.1).
2. Conceptual Model (Sub-section 4.1.3.2.2).
3. Reference process model (in BPMN) for an elective entry case of a patient with a somatic disease (Sub-section 4.1.3.2.3).
4. Application scenario for an emergency entry case of a geriatric patient with a stroke (Sub-section 4.1.3.2.4).

For the standard compliance of the new DSML, the following healthcare standards were considered as additional sources: DefReha© (Gli Ospedali Svizzeri, 2017), International Classification of Functioning, Disability and Health (ICF) standard (World Health Organisation, 2016) and International Statistical Classification of Diseases and Related Health Problems²⁹ (ICD-10).

²⁹ <http://apps.who.int/classifications/icd10/browse/2016/en>

In the following an overview of each of the above-mentioned document type is provided. Sub-section 4.1.3.2.5 describes the requirements elicitation based on the four afore-mentioned documents and the considered standards.

4.1.3.2.1 Information Model

The information model document contains the definitions of all the relevant terms pertaining to the addressed domain, i.e. the transferal management. The document aims to increase a shared understanding among the project stakeholders about the used terminology. Thus, the document was mainly useful for modelling experts and language engineers who are external to the application domain to get familiar with it as well as to identify relevant concepts.

Among others, it was identified that *information documents* and *time* play a crucial role for the transferal management. Fundamentals are documents containing medical and administrative data. These documents are created, modified and exchanged among actors at a certain point in time or within the transferal management process. The process starts in the acute hospital and ends in a rehabilitation clinic. Hence, the information model also presents a rough sketch of the transferal process, including actors and the flow of information documents.

Table 4 shows an excerpt of the document. The complete document can be retrieved in Appendix A: Patient Transferal Management, folder A1.

Table 4. An excerpt of the definition of actors in the research project Patient Radar

Actors	Description
<i>Surgeon (in Acute Hospital)</i>	He/she performs the surgery in the acute hospital. The surgeon also performs the activity of consulting the patient before the surgery, in which the surgeon explains the procedure and suggests the surgery date (e.g. asap or next months). In this activity, no decisions with respect to rehab clinics are made.
<i>Hospital Administrative Personnel (in Acute Hospital)</i>	He/she is responsible for the registration of the patient and opening the patient case. Hence, he/she ensures the correct acquisition of patients' data, e.g. check of the insurance card when patients are hospitalized, informing the health insurance about the patient hospitalization (in the hospital of Walenstadt: this role is covered by employees at the reception).
<i>Transferral Manager (in Acute Hospital)</i>	He/she is responsible for the registration and the transfer of patients from acute hospitals to reha clinics. The transferral manager is in charge of discussing with the patient about his/her rehabilitation as well as selecting together with the patient a rehab clinic. Additionally, he/she interacts with the health insurance for cost reimbursement submission, i.e. the KoGu form., The KoGu form is prepared by the physician, edited by the transferral manager, who then submits it to the health insurance. Hence, the transferral manager must collect that information concerning patient treatments that are relevant for both the health insurance and the rehab clinic. Transferral managers must deal with decisions regarding cost reimbursement. If necessary, the consultant or the case management expert contact the health insurance.
<i>Patient</i>	A patient is hospitalized in an acute hospital. After the patient completes the treatment in the acute hospital, he/she can be transferred to a rehab clinic to receive further rehabilitative treatments.
<i>Bed scheduler</i>	The Bed scheduler arranges and provides the hospitalization date to the patient, opens the case and associates it with a DRG code (von Eiff, Schüring, Greitemann, & Karoff, 2011).
<i>Physician</i>	He/she is responsible for the patient treatment in the acute hospital and for the briefing prescriptions to send to the rehab clinic. Contact information on the physician is included in the briefing prescription and it is available in case of questions/clarifications. He/she is responsible for ensuring that information of the patient questionnaire is transmitted to the transferral manager. The physician does not have to be necessarily the surgeon.

4.1.3.2.2 Conceptual Model

Next, a conceptual model was created based on the information model. This activity addresses the theory-based requirement *TBR5*, which suggests creating a lightweight ontology of the addressed application domain. The lightweight ontology is represented in the form of a conceptual model containing classes, relations and attributes relevant in the transferal

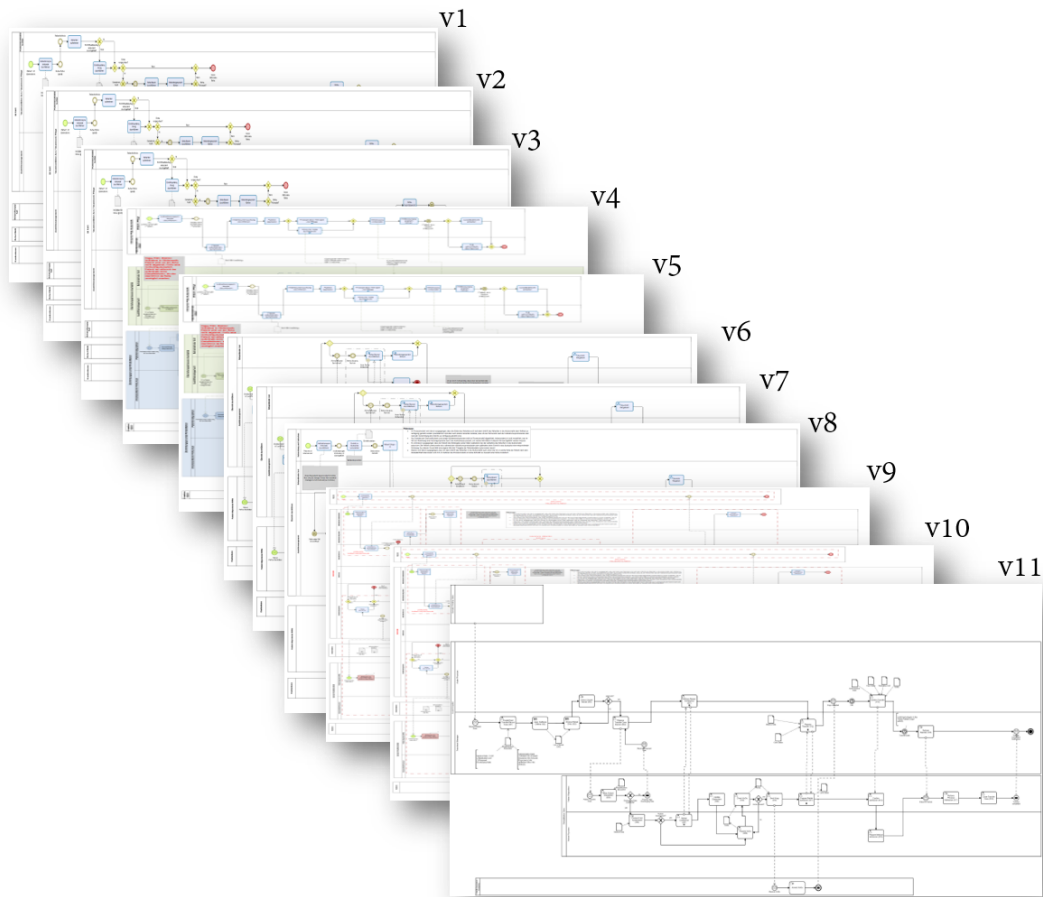


Figure 50. Versions of the Reference Process Models: from the first (v1) to the last version (v11)

The main reason for new changes reverts the two different background of expertise between the language engineers and the transferal managers or physicians. Specially in the early phase of the project, the different background sometimes led to misunderstandings or unclear requirements. As the project advanced, the understanding of the problem deepened, which led to narrowing the scope of the project. At the beginning of the project patients' treatment aspects were considered too, e.g. *surgery activities* as well as the information about the *actor surgeon*. As the project advanced, the focus narrowed to administrative aspects, concerned with the acute hospital. The changes are visible in the reference process models. For instance, see the left-hand side of Figure 51, in which the fifth version of the model is reported. The upper pool of the model shows process activities, events and decisions about clinical pathways in the acute hospital (see red rectangle on the left-hand side of the figure). In contrast, the representation of the process logic in the same pool was clearly not needed anymore in the last version of the reference process model (see right-hand side of Figure 51).

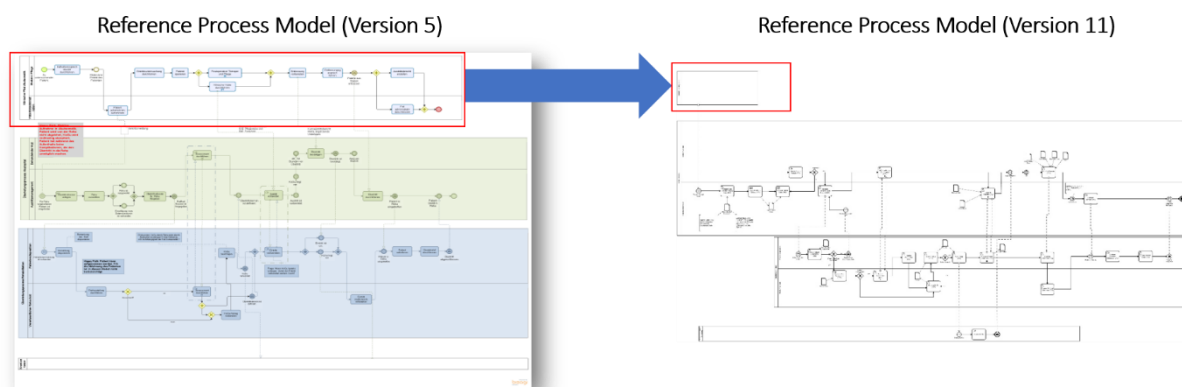


Figure 51. Changes in the Reference Process Model: version 5 vs. version 11

4.1.3.2.4 Application Scenario of an Emergency Entry Case

After the *elective entry case*, another important entry type in transferal management is the *emergency entry case*. An emergency case offers a more interesting scenario than an elective entry case. That is, the administrative set of activities are rather unpredictable (in contrast to the *elective entry cases*) as well as many decisions must be taken under time pressure. Moreover, a DSML that covers these aspects appear more complete with respect to the underlying reality of the transferal management. To deepen understanding on the *emergency entry case*, an application scenario was created.

The application scenario was developed in a workshop conducted by myself with a transferal manager (i.e. a domain expert) from the hospital of Grabs³⁰ (i.e. one of the project partners). The domain expert helped to identify the most common and complex patient case: the *geriatric patient with a stroke*. For this, a descriptive document was developed after the workshop and subsequently approved by the domain expert (see Appendix A: Patient Transferal Management, folder A4). Since the domain expert preferred to conduct the workshop in German, the terminology was translated into English. A cross check was performed between the two so to ensure consistency in the terminology between the document and the above-mentioned information model (Sub-section 4.1.3.2.1). For the new terms, the work in (Walker & Betz, 2013) was considered, as it contains the commonly used terminology in hospital emergency cases.

³⁰ <https://www.srrws.ch/ueber-uns/organisation/spitaeler/spital-grabs.html>

4.1.3.2.5 Requirements Elicitation

This sub-section presents the requirements derived from the above-mentioned document types and standards. Three different sources of group requirements are:

- Requirements from the *application scenario* (Sub-section 4.1.3.2.5.1). As mentioned above, the application scenario focuses on an *emergency entry* case of a geriatric patient with a stroke.
- Requirements from the *reference process model* (Sub-section 4.1.3.2.5.2). As mentioned above, the reference process model focuses on the typical *elective entry case* of a patient with a somatic disease.
- Additional requirements from *workshops* that focused on the elective entry case (Sub-section 4.1.3.2.5.3).

All requirements further consider healthcare standards such as DefReha©, ICF and ICD-10.

Requirements from the both the *application scenario* and the *reference process model* are further divided into four categories:

1. Process requirements,
2. Documents requirements,
3. Information systems and data requirements,
4. Decisions requirements.

Each category is shown in the table containing the number of following *requirements*, a *name of requirement*, one or more *elements to conceptualize* and *knowledge category of requirement*.

In order to address the theory-based requirement *TBR2*, the *knowledge category* should distinguish among: *process logic (structured)*, *process logic (unstructured)*, *business logic (structured)*, *business logic (unstructured)*, *organisational management*, *document management*, *information systems representation*, *data representation*.

4.1.3.2.5.1 Requirements from Application Scenario – Emergency Entry Case

This section presents the requirements derived from the application scenario divided into four tables. Each table focuses on one of the categories mentioned above. The process requirements are shown in Table 5, document requirements in Table 6, information systems and data requirements in Table 7 and decision requirements in Table 8.

Table 5. Process requirements - application scenario

Number	Requirement	Element	Category
RI.1.1	The DSML should accommodate constructs to model specific actors.	Elements reflecting administrative staff, rapid assessment nurse, emergency medical technician, patient, family member, expert nurse, resident physician, specialist physician, transferal manager.	Organisation management
RI.1.2	The DSML should accommodate constructs to model different units.	Elements reflecting Care unit, intensive care unit, non-intensive care unit, hospital, rehabilitation clinic, insurance, emergency room, site of care.	Organisation management
RI.1.3	The DSML should accommodate modelling constructs to model specific activities.	Elements reflecting Perform Physical Admission, Perform transfer, Create Transfer Case Record, Conduct First Assessment ApplyDefReha Criteria, Re-apply DefReha Criteria, Choose Reha Criteria. Release Patient Record, Open Patient Record Close Transfer Case Record, Prepare rehab Admission, Finalize KoGu, Release Transfer Case, Prepare KoGu, Prepare Rehab Admission, Make Disposition of Case, Prepare Medical Record Admission.	Process logic (structured)
RI.1.4	The DSML should accommodate constructs to model a suitable rehabilitation type.	Elements reflecting criteria for Geriatric Rehabilitation Suitability, Internistic and Oncological Rehabilitation Suitability, Neurological Rehabilitation Suitability, Choose Rehabilitation Clinic, Cardiovascular Rehabilitation Suitability, Musculoskeletal Rehabilitation Suitability, Neurological Rehabilitation Suitability, Pediatric Rehabilitation Suitability, Paraplegic Rehabilitation Suitability, Psychosomatic Rehabilitation Suitability, Pneumological Rehabilitation Suitability, Inpatient Rehabilitation Suitability, Compulsory Medical Monitoring Rehabilitation Suitability, Compulsory Medical Monitoring to Inpatient Rehabilitation Suitability	Business logic (structured)
RI.1.5	The DSML should accommodate constructs to model a system.	Elements reflecting the Patient Administration System (PAS), Hospital Information System (HIS).	Information Systems representation
RI.1.6	The DSML should accommodate constructs to model time.	Element reflecting the length of stay.	Data representation
RI.1.7	The DSML should accommodate constructs to model alternative process flow.	Element reflecting alternative process flow are based on health severity, complexity, and admission criteria.	Process logic (structured and unstructured)
RI.1.8	The DSML should accommodate constructs to model a status.	Elements reflecting changing status: medication, indication, care status, diagnosis. General patient status in the hospital and the rehab: patient conditions are improving or worsening.	Data representation
RI.1.9	The DSML should accommodate constructs to model an information exchange.	Element reflecting information exchange among actors: medical data, administrative data, care status from transferal manager to rehab clinic.	Process logic (structured)
RI.1.10	The DSML should accommodate constructs to model condition-based	Elements reflecting activities that are triggered based on conditions. Elements reflecting conditions.	Business logic (structured)

	activities and conditions.		
<i>RI.1.11</i>	The DSML should accommodate constructs to model a reoccurring activity.	Elements reflecting a recurring activity. For instance, re-applying DefRehab criteria.	Process logic (unstructured)
<i>RI.1.12</i>	The DSML should accommodate constructs to model the transfer of a patient.	Element reflecting the transferal of the patient from acute hospital to rehabilitation	Process logic (structured)

Table 6. Document requirements - application scenario

Number	Requirement	Element	Category
<i>RI.2.1</i>	The DSML should accommodate constructs to model the patient admission form.	Element reflecting the hospitalization document that is stored in the Patient Administration System (PAS) and contains administrative data of patient.	Document management
<i>RI.2.2</i>	The DSML should accommodate constructs to model the rehabilitation form for cost reimbursement, i.e. KoGu.	Element reflecting the KoGu form (3.2 of the application scenario) is the cost reimbursement.	Document management
<i>RI.2.3</i>	The DSML should accommodate constructs to model the long report.	The long report contains medical related data, administrative data and investigation results. The transferal manager sends it to the reha clinic.	Document management

Table 7. Information systems and data requirements - application scenario

Number	Requirement	Element	Category
<i>RI.3.1</i>	The DSML should accommodate constructs to model the Hospital Information System (HIS) containing the short report.	Elements reflecting the short report include medical diagnosis, medication list and indication.	Information systems and data representation
<i>RI.3.2</i>	The DSML should accommodate constructs to model the Hospital Information System (HIS) that includes the medical data.	Elements reflecting ICD-10 version 2016 (main), ICD-10 version 2016 (secondary) (several), CHOP, Functional deficits – ICF Standard (several), Flag disease or accident; already allows to choose relevant data for the case.	Information systems and data representation
<i>RI.3.3</i>	The DSML should accommodate constructs to model the Patient Administrative System (PAS) that includes the administrative data/master data.	Elements reflecting Master Patient ID, Patient ID, Case Number, SwissDRG ³¹ code, Name, Street, City, Nation, Nationality, Date of birth, Mobile number, Mother tongue, Job, AHV-Number, Entry Type, Health insurance with policy number, Health insurance status, Date of hospitalization.	Information systems and data representation
<i>RI.3.4</i>	The DSML should accommodate constructs to model the Hospital Information System (HIS) that includes the care status.	Elements reflecting Assistance in Mobility, Assistance in Nutrition, Assistance in Excretion, Assistance in Personal Hygiene, Assistance in Cognition, Necessity of Medical devices, already allows to choose relevant data.	Information systems and data representation
<i>RI.3.5</i>	The DSML should accommodate constructs to model the Hospital Information System (HIS) that includes standard codes that classify the patient problems, i.e. Tessiner code.	Elements reflecting the main and secondary codes that classify the patient's problems.	Information systems and data representation

Table 8. Decision requirements - application scenario

³¹ www.swissdrg.org

Number	Requirement	Element	Category
<i>RI.4.1</i>	The DSML should accommodate constructs to assign the emergency severity to a case (i.e. ESI triage).	Elements to reflecting the level of severity of a case from 1 to 5 (Emergency Severity Index) based on urgency of treatment.	Data representation
<i>RI.4.2</i>	The DSML should accommodate constructs to model the examination conducted by a specialist physician.	Element reflecting the first examination and provides both prescription and diagnosis.	Data representation and document management
<i>RI.4.3</i>	The DSML should accommodate constructs to model the complexity assignment of a case.	Element reflecting the classification of a case as complex or not complex. This depends on the number of diseases/co-morbidities.	Business logic (structured)
<i>RI.4.4</i>	The DSML should accommodate constructs to assign the ICD-10 code to a case.	Element reflecting the ICD-10 code. Every ICD-10 code is associated with a specific length of stay.	Data representation
<i>RI.4.5</i>	The DSML should accommodate constructs to assign the SwissDRG code to a patient case.	Elements reflecting the SwissDRG code: primary and secondary code. The SwissDRG code is calculated at discharging time in the acute hospital and then sent to the insurance. The code calculation is based on both administrative aspects (e.g. age, sex, degree of severity) and medical aspects (e.g. main and secondary diagnosis in terms of ICD-10 code, and the surgery category).	Data representation
<i>RI.4.6</i>	The DSML should accommodate constructs to assign ICF qualifiers.	Elements reflecting three scales of severity for the ICF qualifier: generic qualifier, qualifier for environmental factors, qualifier for body structure.	Data representation
<i>RI.4.7</i>	The DSML should accommodate constructs to model the need or not need for rehabilitation.	Element reflecting the conditions for inclusion or exclusion to rehabilitation.	Business logic (structured)
<i>RI.4.8</i>	The DSML should accommodate constructs to model the decision for the type of rehabilitation clinic and for the specific rehabilitation clinic.	Element reflecting the criteria for choosing a rehabilitation clinic and the type, e.g. main disease of the patient, specialisation of clinic, partner clinic, family member or contact person live close by the clinic.	Business logic (structured)
<i>RI.4.9</i>	The DSML should accommodate constructs to model the activity of checking correctness of KoGu and checking compliance with respect to DefReha© standard.	Elements reflecting the activity of checking KoGu for its consistency with patient's information and compliance with the content of the DefReha©.	Business logic (structured)
<i>RI.4.10</i>	The DSML should accommodate constructs to model the assessment for the admission to a rehabilitation.	Elements reflecting criteria of inclusion and exclusion to a rehabilitation.	Business logic (structured)
<i>RI.4.11</i>	The DSML should accommodate constructs to model the status of the patient getting worse.	Elements reflecting patient conditions getting worse.	Data representation
<i>RI.4.12</i>	The DSML should accommodate constructs to model the status of a patient getting better.	Element reflecting patient conditions getting better.	Data representation

4.1.3.2.5.2 Requirements from Reference Process Model – Elective Entry Case

This sub-section describes the requirements derived from the reference process model. As already mentioned, the reference process model describes the most common entry case of the application domain: the *elective entry case of patients having somatic diseases*. As mentioned already, the different versions of this source led to a continuous change in requirements. The requirements below described refer to the latest version of the reference process model *v11*. Figure 52 depicts the continuous intertwin required between the domain analysis and the requirement elicitation activity.

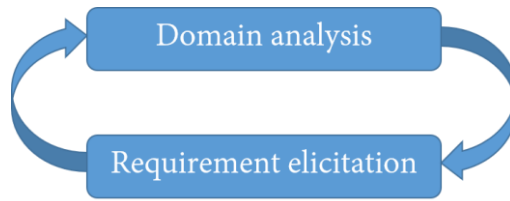


Figure 52. Intertwin between the domain analysis and the requirement elicitation activities

In a later stage of the project, new requirements had to be accommodated too. It was requested to create an executable workflow for the reference process model. This led to create a different level of abstraction for the reference process model. For instance, a distinction between activity types was needed, i.e. manual task, service task, business rule task. Additionally, each process activity was analysed in detail to define the input and output data. Hence, for each process activity, one or more mock-ups were created as well as validated by domain experts.

Therefore, the mock-ups were considered as an additional source of requirement (see the additional list of requirements in Sub-section 4.1.3.2.5.3).

A total of 47 mock-ups were created. One mock-up is depicted in Figure 53 and shows the process activity “*Prepare KoGu*”, i.e. prepare cost reimbursement activity. The document containing all the mock-ups can be found in Appendix A: Patient Transferal Management, folder A6.

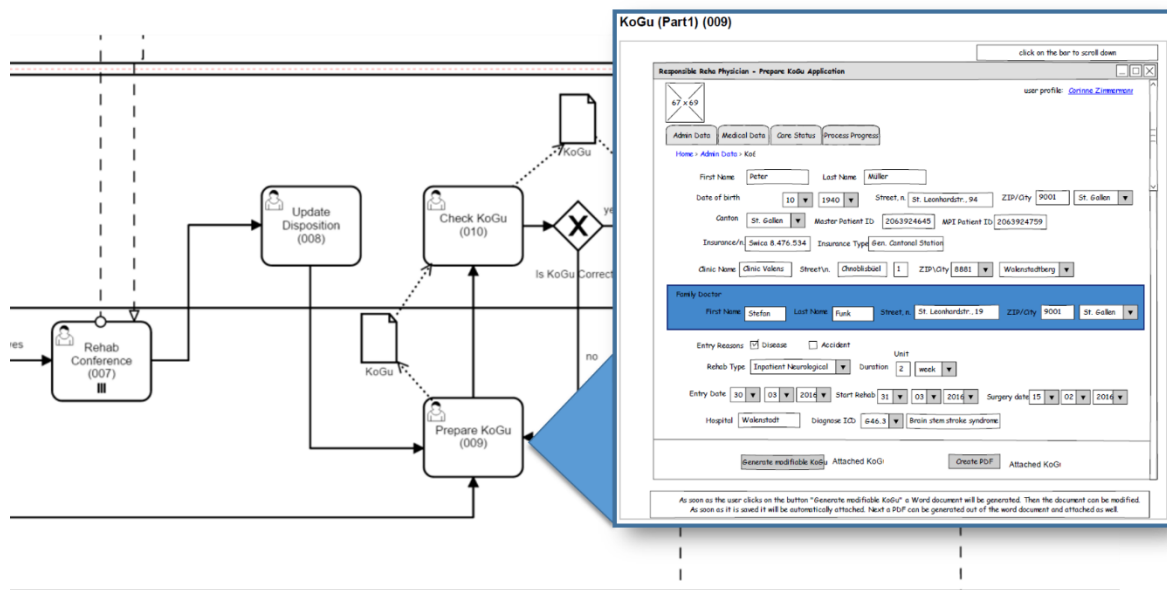


Figure 53. Mock-up for the process activity “*Prepare KoGu*”

A total of 18 process activities were identified for the reference process. Activities are assigned to a sequence of numbers, which reflects the execution flow in the process. Figure 54 shows all the 18 activities. The document source can be found in Appendix A: Patient Transferal Management, folder A7.

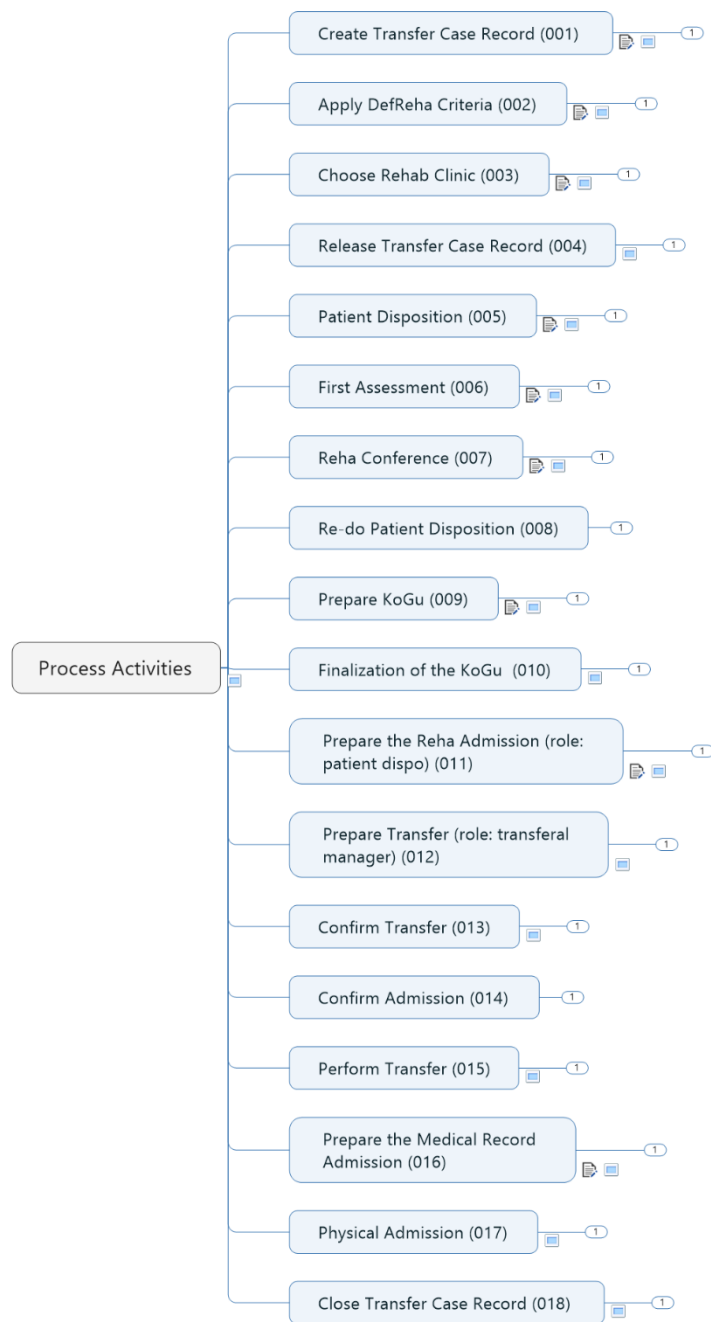


Figure 54. Reference Process Model Activities

Below, the requirements derived from the reference process model are grouped in four tables: process requirements in Table 9, document requirements in Table 10, information systems and data requirements in

Table 11, decision requirements in Table 12.

Table 9. Process requirements - reference process model

Number	Requirement	Element	Category
R2.1.1	The DSML should accommodate constructs to model different actors.	Elements reflecting actors: Physician, exit management, patient disposition, rehabilitation physician.	Organisational management
R2.1.2	The DSML should accommodate constructs to model different units/processes.	Elements reflecting units: Acute hospital, rehabilitation clinic.	Organisational management
R2.1.3	The DSML should accommodate constructs to model activities.	Elements reflecting generic and specific activities.	Process logic (structured and unstructured)
R2.1.4	The DSML should accommodate constructs to model a decision/business rule activity.	Elements reflecting decisions/rules.	Business logic (structured)
R2.1.5	The DSML should accommodate constructs to model parallel activities.	Element reflecting the possibility to execute two task elements in parallel.	Process logic (structured and/or unstructured)
R2.1.6	The DSML should accommodate constructs to model manual activity.	Element reflecting a manual task.	Process logic (structured and/or unstructured)
R2.1.7	The DSML should accommodate constructs to model time.	Element reflecting time, e.g. time until transfer.	Process logic (structured and/or unstructured)
R2.1.8	The DSML should accommodate constructs to model separate paths (AND and XOR).	Elements reflecting paths: select rehabilitation clinic, first assessment, prepare entry.	Process logic (structured)
R2.1.9	The DSML should accommodate constructs to model information.	Elements reflecting information to be sent to other actors (e.g. Hospitalization document).	Data representation
R2.1.10	The DSML should accommodate constructs to model the transfer of a patient.	Element reflecting the need to transfer a patient from acute hospital to rehabilitation.	Process logic (structured)
R2.1.11	The DSML should accommodate constructs to model the activity rehab conference, which can start at any time on request of the rehab physician	Element reflecting the execution of a task on request and not by following a prescribed flow.	Process logic (unstructured)

Table 10. Document requirements - reference process model

Number	Requirement	Element	Category
R2.2.1	The DSML should accommodate constructs to model the patient admission form.	Element reflecting administrative data of patient.	Document management and data representation
R2.2.2	The DSML should accommodate constructs to model the rehabilitation form for cost reimbursement (KoGu).	Element reflecting the cost reimbursement (KoGu) and the relevant data of it.	Document management and data representation

Table 11. Information systems and data requirements - reference process model

<i>Number</i>	Requirement	Element	Category
R2.3.1	The DSML should accommodate constructs to model the Hospital Information System (HIS) that includes medical data.	Elements reflecting system element and related medical data.	Data representation
R2.3.2	The DSML should accommodate constructs to model the Patient Administrative System (PAS) that includes administrative data.	Elements reflecting system element and related administrative data.	Data representation
R2.3.3	The DSML should accommodate constructs to model the Hospital Information System (HIS) and the Patient Administrative System (PAS) that include documents.	Elements reflecting system elements with applicable attributes and/or data/document elements.	Document management and data representation

Table 12. Decision requirements - reference process model

<i>Number</i>	Requirement	Element	Category
R2.4.1	The DSML should accommodate constructs to model the creation of the transferal case.	Elements reflecting criteria for business rules.	Business logic (structured)

4.1.3.2.5.3 Additional Requirements

Additional requirements were elicited from the above-mentioned mock-ups that was a new source. In addition, the standard DefReha© had to be further considered as well as additional requirements derived from project workshops.

On the one hand, mock-ups mainly supported the detailed specification of documents and data elements, status elements and relevant attributes.

On the other hand, the DefReha© standard defines the inclusion and exclusion criteria for nine different rehabilitation types. Each criterion corresponds to a set of rules that leads either to admit or to discharge a patient to or from a rehabilitation clinic. The execution of these rules should be used to support the transferal manager on the choice of both the type and the rehabilitation clinic. The additional requirements are reported in Table 13.

Table 13. Additional requirements

Number	Requirement	Element	Category	Source
R3.1.1	The DSML should accommodate constructs to model organisation and roles in a hierarchical way.	Elements are allowed to view the involved organisations, roles and actors in a hierarchical way.	Organisational management	Project workshop
R3.1.2	The DSML should accommodate constructs to allow automation in a later stage.	Elements reflecting data/documents should be filled in with patient data.	Document and data representation	Project workshop
R3.1.3	The DSML should accommodate constructs to reflect the different rehabilitation types according to the DefReha© standard.	Elements reflecting rehabilitation types (possibility for the user to choose it).	Data representation	DefReha© Standard
R3.1.4	The DSML should accommodate constructs to model criteria according to the DefReha© standard.	Elements reflecting entry, exit, inclusion and exclusion criteria for the different rehabilitation types, i.e. rules defined in the DefReha©.	Business logic (structured)	DefReha© Standard
R3.1.5	The DSML should accommodate constructs to reflect attributes corresponding to the attributes of the mock-ups.	Element reflecting attributes for all documents, data, groups and status elements.	Data representation	Mock-ups
R3.1.6	The DSML should accommodate constructs to model documents/data elements corresponding to the mock-ups.	Elements reflecting the following documents/data objects with hierarchical visualization: Administrative Data with sub elements Patient Data, Patient Health Insurance Data, Acute Physician Data, Rehabilitation Data, KoGu, Hospitalization Document, Medical Data with sub elements Medical Information, ICF Standard, Medication List, Care Status with sub elements Assistance Data, Special Medication Data, Process Progress with sub element Process Data.	Document management and data representation	Mock-ups
R3.1.7	The DSML should accommodate constructs to model relevant status elements and their attributes corresponding to the mock-ups.	Elements reflecting the following status elements with hierarchical visualization: Status with sub elements Process Status, Physical Transfer Status, KoGu Status, ICF Qualifiers Status, Medication List Status and Acceptance Status.	Data representation	Mock-ups
R3.1.8	The DSML should accommodate constructs to model KoGu templates.	Elements reflecting the KoGu templates: Stroke template, Orthopaedic template, Paediatric template, Psychiatric template.	Data representation.	Project workshop
R3.1.9	The DSML should accommodate constructs to differentiate between tasks performed by a user or by a system.	Elements reflecting user tasks and service tasks. The latter allows automating the transferal management process.	Process logic (structured and/or unstructured)	Project workshop
R3.1.10	The DSML should accommodate constructs to execute activities based on conditions	Elements reflecting tasks triggered by conditions. This is needed in case the document KoGu gets rejected. For instance, the DefReha criteria will need to be re-applied.	Process logic (unstructured)	Project workshop

4.1.3.2.6 Conclusion and Problems in the Create phase

The elicitation of the application domain-based requirements addressed the theory-based requirement *TBR2* by indicating the different knowledge categories: process logic (structured), process logic (unstructured), business logic (structured), business logic (unstructured), organisational management, document management, information systems representation or data representation.

Given the different identified categories of knowledge (e.g. business logic, data representation etc.) other modelling languages should be considered. This activity is shown in the design phase of the AMME Lifecycle (Sub-section 4.1.4) fulfils the theory-based requirement *TBR4*.

Additionally, from the domain analysis the need to execute models created with the DSML4PTM was identified. For example, execution of business rule tasks like re-apply DefReha©, which determines the suitable type of rehabilitation for a patient as well as displaying the progress of an indicator according to some aggregated data. As reported in the theory-based requirement *TBR1*, a DSML should enable domain experts to design, adapt and use models and this is also the case of DSML4PTM.

Finally, the above-reported experience in the Creation phase led to identify three major problems that can hinder agility in DSML engineering:

***Problem 1:** Language engineers and domain experts have different types of expertise. This might result in misinterpretation of requirements. As a consequence, the quality of the new version of DSML is hampered. Other engineering iterations are required as well. This problem manifests particularly in the beginning of the project. As the language engineers have little knowledge about the addressed domain.*

***Problem 2:** The extraction, documentation, prioritization, and categorization of requirements is a time-consuming manual task. All requirements were categorised, documented, and prioritized. The more the requirements, the more time consuming this task gets.*

***Problem 3:** The update and synchronization of requirements is a time-consuming task. Although it is a good practice, it slows down the engineering of the DSML. Change of requirements led to update and synchronize requirements on each other. The synchronization was performed to ensure consistency among requirements that are dependent with each other. Especially in the early stage of the project, requirements changed frequently, which made it difficult to quickly move on to the subsequent phase (Design phase). The longer the list of requirements and the more the dependencies among requirements, the more time-consuming this update and sync.*

The next sub-section describes the design of the meta-model of DSML4PTM, which addresses the elicited requirements.

4.1.4 Design Phase

In this phase, a tailored procedure for the development of DSML4PTM was initially suggested. Next, the design of the meta-model is presented, which addresses all the requirements elicited in the previous phase.

As suggested by the theory-based requirement *TBR4* in Sub-section 4.1.3.1.5, a procedure should be followed as per the rigorous design of the meta-model. The procedure proposed in (Braun et al., 2016) and derived from (Stroppi et al., 2011) could be eligible for this purpose.. However, the procedure is limited to the extension of BPMN only. In our case, several modelling languages should be selected, and where required, concepts from different languages should be integrated with one another, for example, the integration between structured and unstructured process logic performed in (Hinkelmann, 2016). The procedure followed by Burwitz et al. (2013) seems to be of a suitable basis for our work. It foresees the direct customization of the meta-model and integration among concepts that belong to a different language. We build on top of this procedure, to provide more thoroughness in the construction of the DSML.

The proposed procedure finds its fundamentals in two principles suggested in (Karagiannis et al. 2016):

- The inclusion of existing standard modelling languages, with related applications and lessons learned.
- The specialisation of language constructs according to requirements elicited from a specific domain.

As Figure 55 shows, the procedure is regarded as a concrete instantiation of the Design phase of the AMME Lifecycle.

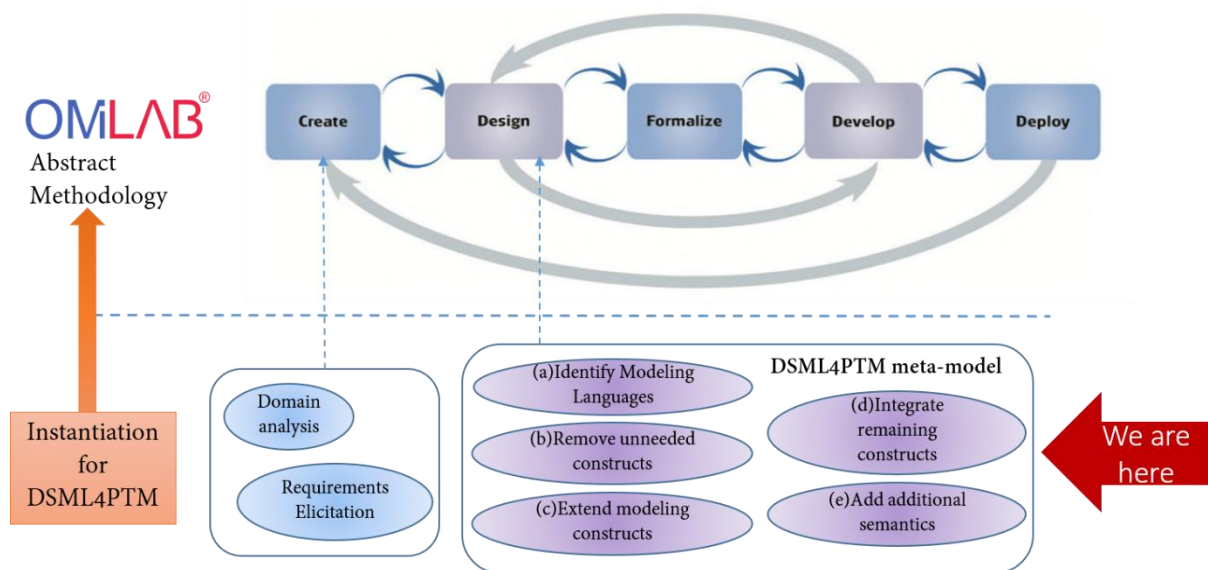


Figure 55. Instantiation of the Design phase: the DSML4PTM meta-model

The violet light bubbles in Figure 55 (from (a) to (e)) show the suggested procedure steps. The five steps are performed according to the elicited application domain-based requirements (ADBRs):

- (a) Identify the most suitable existing modelling languages.
- (b) Remove the unneeded modelling elements (as Silver (2011) suggests) from the selected languages.
- (c) Identify modelling constructs to be extended and extend them.
- (d) Integrate the language constructs that belong to different modelling languages. It also includes bridging connectors between elements of different meta-models.

- (e) Add additional semantics in the form of constraints among the remaining language constructs.

Steps (b) and (c) can also be inverted. These steps were embedded into a two-tier approach that refers to the meta-modelling hierarchy introduced in Section 2.3.

(a), (b), (c), and (d) are all part of the abstract syntax, which resides in level 2 of the meta-model stack (see Figure 56). They all contain semantics. The identified meta-models in (a), for example, already contain their semantics. Moreover, while for (b) semantics is removed, it is added for (c). Integrating model elements with each other also implies additional semantics, (see (d) in Figure 56). Additional semantics is added in terms of new relations among meta-model elements, (see (e) in Figure 56).

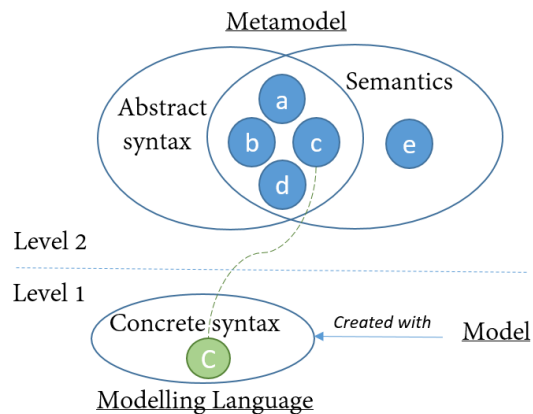


Figure 56. Steps to design the meta-model of DSML4PTM

Similar concept is presented by Laurenzi (2014) and Reimer and Laurenzi (2014), the two-tier approach allows the following:

- *On Level 1*, administrative pathways from acute hospitals and rehabilitation clinics are combined into a coherent discharging process. New processes can be accommodated, or the existing ones adapted as the discharging process that fits one acute hospital might differ for another clinic.
- *On Level 2*, the new meta-model can be extended further to cover new domain requirements, e.g. transferring patients from acute hospitals to nursing facilities.

In total, the DSML4PTM meta-model consists of about 100 concepts and 300 attributes. In contrast to single modelling language extensions, all the unneeded concepts were removed. Thus, the meta-model remains lean and tailored for the targeted application domain.

According to the different *knowledge types* that were identified in the requirement elicitation, five distinct and interconnected modelling views were proposed:

1. A process modelling view, which is the main view (Section 4.1.4.1).
2. A control element modelling view (Section 4.1.4.2).
3. A decision modelling view (Section 4.1.4.3).
4. A document and knowledge modelling view (Section 4.1.4.4).
5. An organisational view (Section 4.1.4.5).

Below, each modelling view is motivated with respect to the gathered requirements, and for each view a suitable meta-model is chosen (Step (a) of Figure 55). Next, elements of the chosen meta-model are kept or extended (Step (c) of Figure 55) or they are removed (Step (b)

of Figure 55) according to the language requirements. Also, the integration among concepts of different meta-models (Step (d) of Figure 55) is shown along with the introduction of meta-models. Next, additional semantics is added (Step (e) of Figure 55) for the extended modelling constructs.

4.1.4.1 Process Modelling View

According to the elicited requirements, there is the need to represent structured and unstructured process logic (category of knowledge type). Some activities and conditions are known in advance while the execution of others depends on human judgment or external events. Thus, it is required to combine predefined, clearly structured process parts with more ad hoc process steps.

The reference process depicting the *elective entry case* is mainly characterized by a prescribed flow of activities. For example, the patient's disposition in the rehab clinic cannot start before the transfer manager releases all the necessary data and documents related to the patient (see Figure 57).

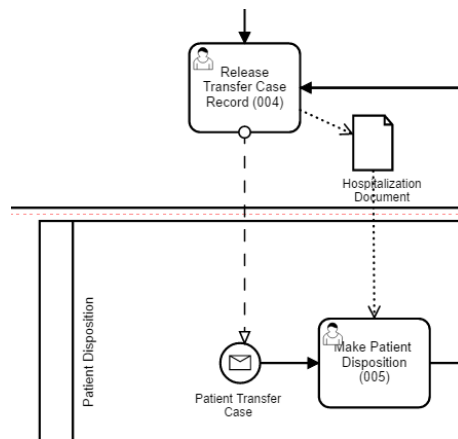


Figure 57. Example of prescribed flow in the reference process

In contrast, the responsible physician from the rehabilitation clinic might discuss with the main physician from the acute hospital about the patient's therapy. The activity execution of this activity is up to the responsible physician rather than dictated by a sequence flow (e.g. see requirement *R2.1.11* in Sub-section 4.1.3.2.5.2).

4.1.4.1.1 BPMN

As above-motivated BPMN 2.0 was selected to model activities and conditions, which are known in advance and their flow can thus be modelled. In addition, unnecessary BPMN elements were removed and others were extended with new elements to satisfy our list of requirements (see concepts in light blue bubbles in Figure 58).

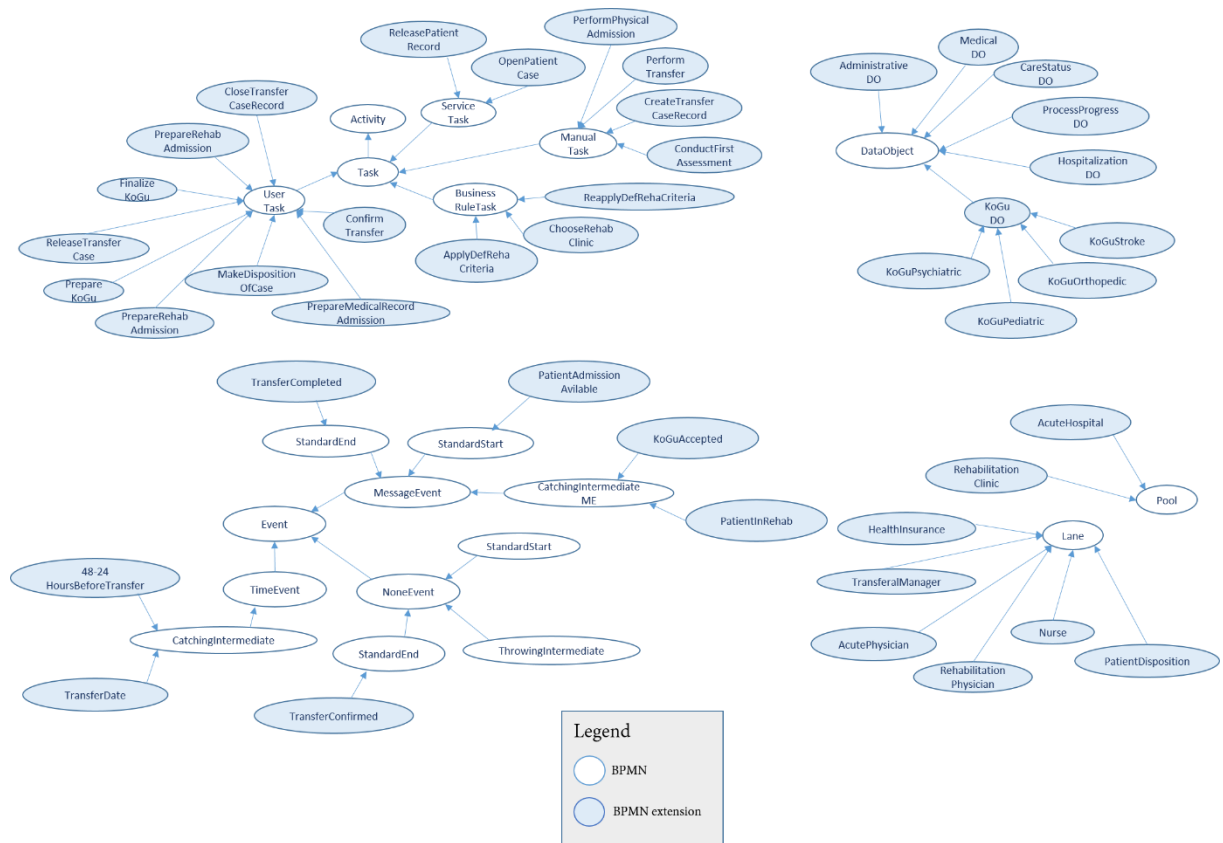


Figure 58. Extended BPMN

A more detailed list of elements is reported as follows:

- **Activities:** The concept “Task” and its specification (“Manual”, “User”, “Service” and “Business Rule”) are needed. Extended elements are as follows:
 - **Manual Task:** “PerformPhysicalAdmission”, “PerformTransfer”, “CreateTransferCaseRecord”, “ConductFirstAssessment”.
 - **User Task:** “ConfirmTransfer”, “PrepareMedicalRecordAdmission”, “MakeDispositionOfCase”, “PrepareRehabAdmission”, “PrepareKoGu”, “ReleaseTransferCase”, “FinalizeKoGu”, “PrepareRehabAdmission”, “CloseTransferCaseRecord”.
 - **Service Task:** “ReleasePatientRecord”, “OpenPatientCase”.
 - **Business Rule Task:** “ApplyDefRehaCriteria”, “ChooseRehabClinic”, “ReapplyDefRehaCriteria”.
- Tasks need to be linked with “Sequence Flow”. “Task” can be marked as “Loop” and “Parallel”. Other concepts regarding the concept activity are removed from the BPMN meta-model.

- *Gateways*: The concepts “Exclusive Gateway” and “Parallel Gateway” are needed. Other concepts regarding gateways are removed from the BPMN meta-model.
- *Events*: The red marked elements in Figure 59 depicts the required events.















Types	Start		Intermediate				End
	Top-Level		Catching	Boundary Interrupting	Boundary Non-Interrupting	Throwing	
None							
Message							
Timer							
Error							

Figure 59. Relevant events according to requirements. Adapted from (Silver, 2011).

The selected events will be implemented to be selected by the user. Domain-specific events are included in the meta-model as subclasses of “Event”. The rest of the events are removed from the BPMN meta-model.

- *Data*: The concept “Data Object” is needed. It may represent both data and documents. “Data Object” is extended with the following concepts: “KoGu”, “Medical Data”, “Administrative Data”, “Process Progress”, “Care Status” and “Hospitalization Document”. Data objects is linked with “Data Associations”. The requirement for the process progress was addressed by adding a new concept “*Status*”. This includes the percentage attribute that reflects the actual state of the process (see the related concrete syntax in the point 6 of Figure 71). The rest of the concepts regarding data object are removed from the BPMN meta-model.
- *Swim lanes*: All concepts regarding swim lanes are needed. Namely, “Pool”, “Lane” and “Message Flow”. The “Pool” concept is extended with “Acute Hospital”, “Rehabilitation Clinic” and “Health Insurance” pools. The “Lane” concept is extended with “Transferal Manager”, “Acute Physician”, “Rehabilitation Physician”, “Nurse” and “Patient Disposition” lanes.
- *Choreography/Collaboration/Conversation*: No concepts to model a choreography, a collaboration or a conversation are needed. Hence, choreography, collaboration and conversation concepts are removed from the BPMN meta-model.

Table 14 shows an overview of the BPMN concepts needed by the requirements listed in the first column.

Table 14. Overview of the covered requirements

Req. Number	Concepts for the BPMN meta-model
<i>R1.1.1-R1.1.4, R1.1.6-R1.1.9, R1.2.2, R1.4.1-R1.4.10, R2.1.1-R2.1.9, R2.2.2, R3.1.3, R3.1.5-R3.1.7-R3.1.9</i>	Task; Manual Task: PerformPhysicalAdmission, PerformTransfer, CreateTransferCaseRecord, ConductFirstAssessment; User Task: ConfirmTransfer, PrepareMedicalRecordAdmission, MakeDispositionOfCase, PrepareRehabAdmission, PrepareKoGu, ReleaseTransferCase, FinalizeKoGu, PrepareRehabAdmission, CloseTransferCaseRecord"; Service Task: ReleasePatientRecord, OpenPatientCase; Business Rule Task: ApplyDefRehaCriteria, ChooseRehabClinic, ReapplyDefRehaCriteria; Loop Marker; Parallel Marker; Sequence Flow; Exclusive Gateway; Parallel Gateway; Pool: Acute Hospital, Rehabilitation Clinic, Health Insurance; Lane: Transferal Manager, Acute Physician, Rehabilitation Physician, Nurse, Patient Disposition; Message Flow; Data Object: KoGu, Medical Data, Administrative Data, Process Progress, Hospitalization Document, Care Status; Data Association; None Event: Standard Start, Throwing Intermediate, Standard End; Message Event: Standard Start, Catching Intermediate, Throwing Intermediate, Standard End; Timer Event: Catching Intermediate; Status.

4.1.4.1.2 CMMN

In order to model the activities and conditions that cannot be embedded in the process flow, CMMN (OMG, 2016a) was chosen (i.e. the OMG standard for case management modelling). Again, we removed the unneeded elements as well as extended new ones. “Sentry” and “Entry Criterion” were kept, while for “Discretionary Task” specialisation like *Update Disposition* and *Perform Rehab Conference* were introduced.

The integration with the BPMN elements was done by connecting the sentry to the task. According to our proposed semantics, a task can be performed either as a subsequent activity as part of a flow, or as soon as the sentry evaluates to true. Moreover, a task can have one or more sentries. Two or more sentries express the “OR” condition. Thus, if one sentry evaluates to true, the token steps into the task. The same also applies in presence of one input flow and at least one sentry.

The discretionary task concept is a sub-class of the manual task (see the bubbles with blue outline in Figure 60). At run-time, discretionary tasks that are involved in the sequence flow are skipped if none of the attached sentry evaluates to true. For example, the *PerformRehabConference* is a discretionary task that can be executed by the rehab if the patient case is complex or simply if he or she wants to discuss the case with the physician in the acute hospital.

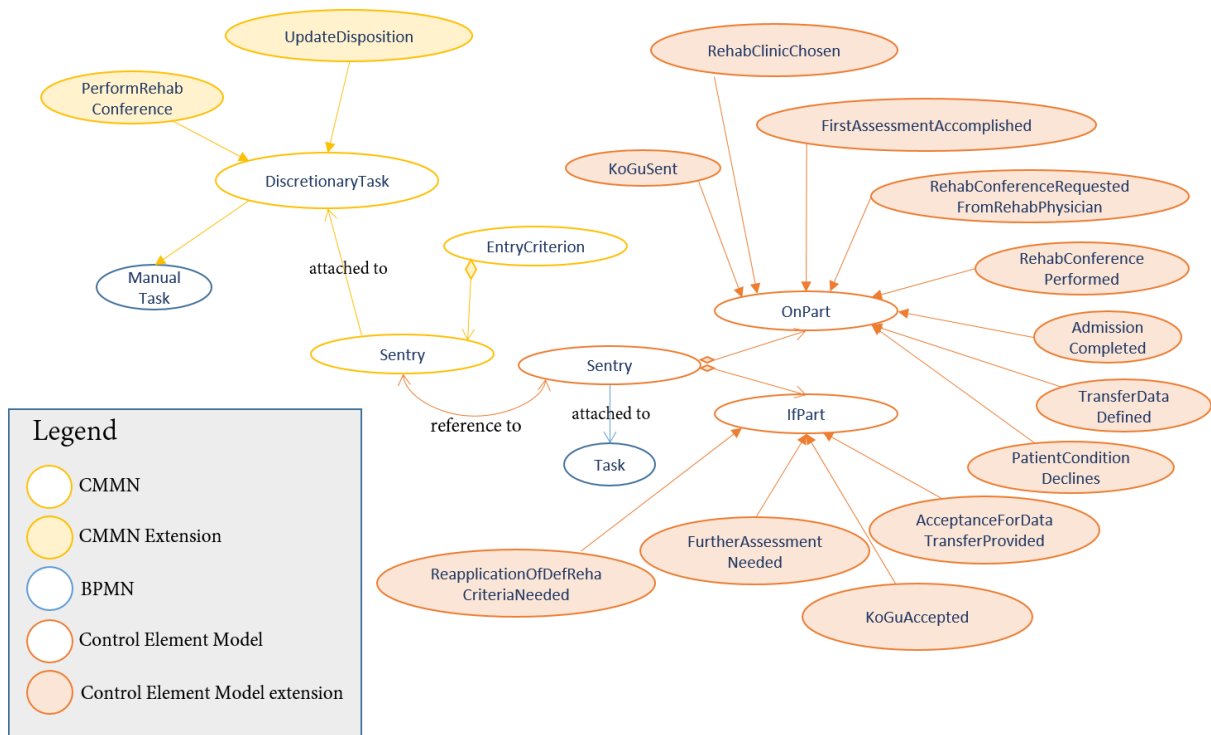


Figure 60. Extended CMMN, extended Control Element Meta-Model and their relation

4.1.4.2 Control Element Modelling View

Complex decisions are made along with the discharging process. For example, the acute hospital decides on whether to start performing the admission for the incoming patient (e.g. preparing all the needed resources). Such decision can be made only after the transferal date has been agreed upon between the acute hospital and rehabilitation clinic and after the cost reimbursement form has been sent to the health insurance. Another example is about decisions that require to be taken if the patient’s situation worsens.

As stated in (Hinkelmann 2016), sentries can be specified in the so-called Control Element Model in order to enable reuse of conditions and events. The metamodel elements that are considered from the Control Element Model are “On-Part” and “If-Part”. Figure 60 shows the extended elements in the light orange bubbles as well as their integration with the sentry element of the CMMN.

Table 15 shows an overview of the concepts in both meta-models CMMN and Control Element Model that are needed to fulfil the requirements listed in the first column.

Table 15. Overview of the needed concepts from CMMN and Control Element Meta-Model

<i>Req. Number</i>	Concepts for the CMMN meta-model and for the Control Element meta-model
<i>R1.1.10-R1.1.11,</i> <i>R1.4.11-R1.4.12</i>	CMMN: Discretionary Task, Sentry, Entry Criterion
<i>R2.1.11-R3.1.10</i>	Control Element Model: Sentry, Connection, On-Part, If-Part

4.1.4.3 Decision Modelling View

The decision logic for complex decisions along with the discharging process must be modelled on the activity level. For example, the application of the right discharging criteria permits to derive the most suitable rehabilitation type and interface (e.g. from acute hospital to inpatient neurological rehabilitation, rather than to a rehabilitation with compulsory medical monitoring). The output is then used as input for another set of rules to identify suitable rehabilitation clinics. Figure 61 depicts this scenario as an excerpt of the reference process.

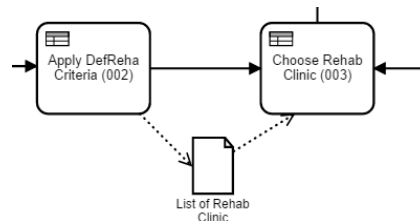


Figure 61. Example of decisions logic modelled in the reference process

In Switzerland, discharging criteria are specified in the DefReha[®] standard issued by the organisation H+ Swiss Hospital (Gli Ospedali Svizzeri, 2017).

In order to model decision logic we selected the DMN standard (OMG, 2016b). The metamodel concerning the Decision Requirements and the Decision Logic were reused and integrated into the metamodel of the new DSML. The analysis of the DefReha[®] document has shown, that the decision to determine the rehabilitation category and relevant inclusion and exclusion criteria are of central importance for the domain in Switzerland. Figure 62, shows some of relevant extended elements in light green bubbles (the complete list is shown below). The integration took place via the decision element, which refers to the business rule task and the discretionary task (see the two concepts connected to decision on the bottom left part of Figure 62).

Additional constraints were added among the extended elements. In Section 4.1.5 an example about how the DMN elements are related with each other is described.

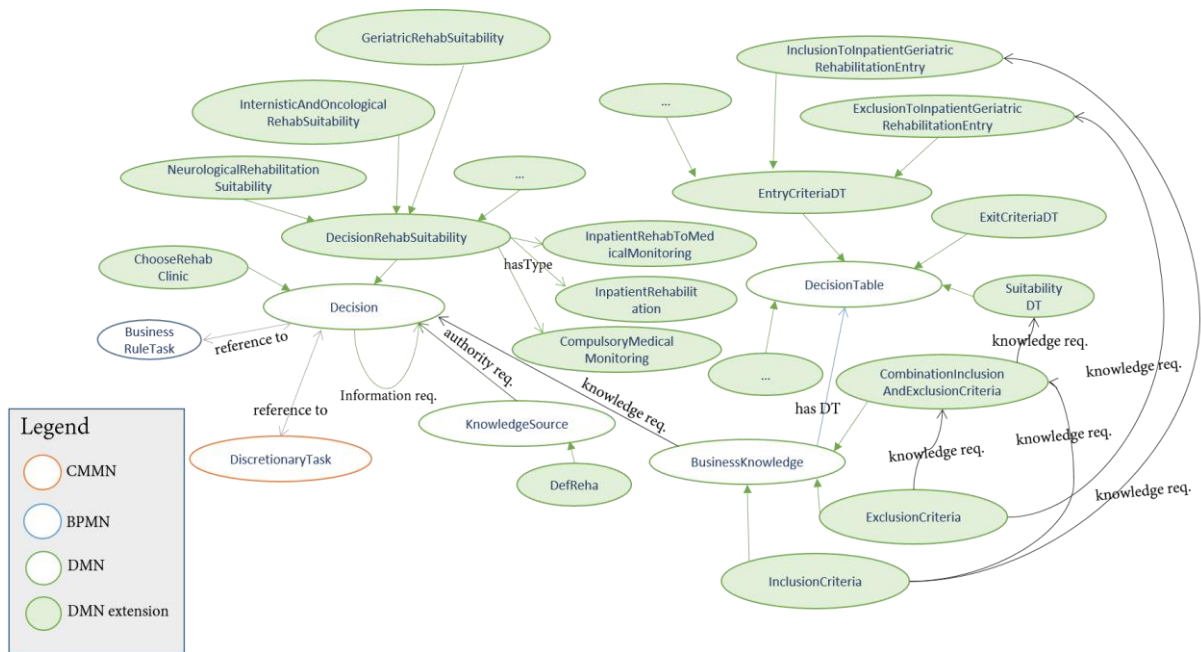


Figure 62. Extended DMN and references with other meta-models

The complete list of extended elements is following listed:

- *Decision*: “Choose Rehab Clinic”, “Geriatric Rehabilitation Suitability”, “Internistic and Oncological Rehabilitation Suitability”, “Cardiovascular Rehabilitation Suitability”, “Musculoskeletal Rehabilitation Suitability”, “Neurological Rehabilitation Suitability”, “Paediatric Rehabilitation Suitability”, “Paraplegic Rehabilitation Suitability”, “Psychosomatic Rehabilitation Suitability”, “Pneumological Rehabilitation Suitability”, “Inpatient Rehabilitation Suitability”, “Compulsory Medical Monitoring Rehabilitation Suitability”, “Compulsory Medical Monitoring to Inpatient Rehabilitation Suitability”.
- *Business Knowledge*: “Exclusion Criteria”, “Inclusion Criteria”, “Combination Inclusion and Exclusion Criteria”.
- *Decision Table*: “Entry Criteria Decision Table”, “Exit Criteria Decision Table”, “Suitability Decision Table”, “Inclusion to Inpatient Geriatric Rehabilitation Entry”, “Inclusion to Inpatient Geriatric Rehabilitation Exit Home/Long-term Facility”, “Exclusion to Inpatient Geriatric Rehabilitation Entry”, “Exclusion to Inpatient Geriatric Rehabilitation Exit”.
- Knowledge Source: “DefReha©”.

The listed concepts allow to model all possible decisions of the DefReha©. This implies that criteria for decision tables like “Entry Criteria Decision Table”, “Exit Criteria Decision Table”, are entered at the time of their instantiation.

However, for elective entry cases we learned that specific decision tables for geriatric patient cases can be useful for the transferal manager. The advantage would have been to have this decision table already available with specific criteria. Thus, for the decision *Geriatric Rehabilitation Suitability*, predefined decision tables were created. These are above emphasised in italics style, e.g. *Inclusion to Inpatient Geriatric Rehabilitation Entry*. The result in terms of concrete syntax is shown in the development.

Table 16 shows the overview of needed concepts from the DMN meta-model as well as the extended modelling elements.

Table 16. Overview of the needed concepts from the DMN meta-model

Req. Number	Concepts for the DMN meta-model
<i>R1.1.4, R1.4.1, R1.4.7- R1.4.10, R2.1.4, R3.1.4</i>	Decision: Choose Rehab Clinic, Geriatric Rehabilitation Suitability, Internistic and Oncological Rehabilitation Suitability, Cardiovascular Rehabilitation Suitability, Musculoskeletal Rehabilitation Suitability, Neurological Rehabilitation Suitability, Pediatric Rehabilitation Suitability, Paraplegic Rehabilitation Suitability, Psychosomatic Rehabilitation Suitability, Pneumological Rehabilitation Suitability, Inpatient Rehabilitation Suitability, Compulsory Medical Monitoring Rehabilitation Suitability, Compulsory Medical Monitoring to Inpatient Rehabilitation Suitability, Knowledge Source: DefReha©; Input Data; Business Knowledge: Exclusion Criteria, Inclusion Criteria, Combination Inclusion and Exclusion Criteria; Decision Table: Entry Criteria Decision Table, Exit Criteria Decision Table, Inclusion to Inpatient Geriatric Rehabilitation Entry, Inclusion to Inpatient Geriatric Rehabilitation Exit Home/Long-term Facility, Exclusion to Inpatient Geriatric Rehabilitation Entry, Exclusion to Inpatient Geriatric Rehabilitation Exit. Relations: Information Requirement; Knowledge Requirement; Authority Requirement; Has Decision Table.

4.1.4.4 Documents and Knowledge Modelling View

The modelled documents have to comply with healthcare standards like the International Classification of Functioning, Disability and Health (ICF) standard (World Health Organisation, 2016).

Representing the relevant documents and data is also required. For example, the cost reimbursement form (abbreviated as KoGu) is the most important document in the transferal management process. In case it is rejected, the discharging process most likely comes to an end due to the lack of financial support.

Another essential document is the hospitalization form, which contains all the needed information for the transferal manager to start the process. The process progress along with the patient hospitalization should provide visibility to crucial events such as case and patient accepted by the clinic, first assessment performed, discharging date agreed. Then, the medication list form that must be updated after the transferal is confirmed.

Finally, each document belongs to a category. For instance, the hospitalization document belongs to the administrative data, the medication list belongs to the medical data, whereas the assistance list belongs to the care status data.

There is also the need to represent the status and versions of the document. For example, the KoGu document should include the status (i.e. ready, sent, rejected or accepted), while the ICF document should include both the trend and the status of each category.

Complying with many standards as well as accommodating many relevant documents were addressed by selecting the Documents and Knowledge Model adopted in the European project LearnPAd (De Angelis et al., 2016). This provides a way to structure documents and knowledge representations. For example, we introduced the new concept “KoGu data object” as a specialised data object. The required status of the KoGu data object was modelled by means of attributes (Figure 63 shows the concrete syntax of the KoGu data object with related status as well as the hospitalization document with related version on top of the icon).

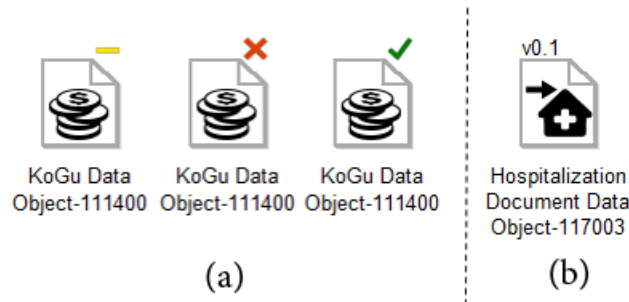


Figure 63. Example of status and version on extended data objects

Constraints for defining the structure of data and documents were also added. Due to the very large collection of data and documents, we grouped them into *medical data*, *administrative data*, *care status data* and *process progress*. Each group contains several data and documents (e.g. see assistance data and special medication data connected with the care status in Figure 64).

Additional constraints addressed the need for indicating the status of data or documents. For instance, the metamodel contains a connection between the *ICF standard* and the *ICF qualifier status*, and between the *KoGu document* and its *status* (see Figure 64). The same applies on other four elements, i.e. *process status*, *physical transfer status*, *medication list status*, and *acceptance status*. These “Status” elements are determined based on the progress of the data collection and are used to aggregate the overall “Status” in the DSML4PTM model.

Bubbles with light yellow in Figure 64 depict some of the specialised elements, while connections to other coloured bubbles show their reference to other meta-model elements, i.e. from BPMN (in light blue) and DMN (in light green). This reference relation allows the navigability to a different diagram (or views) at model level.

Table 17 shows the needed concept from the document and knowledge meta-model such that correspondent requirements are fulfilled.

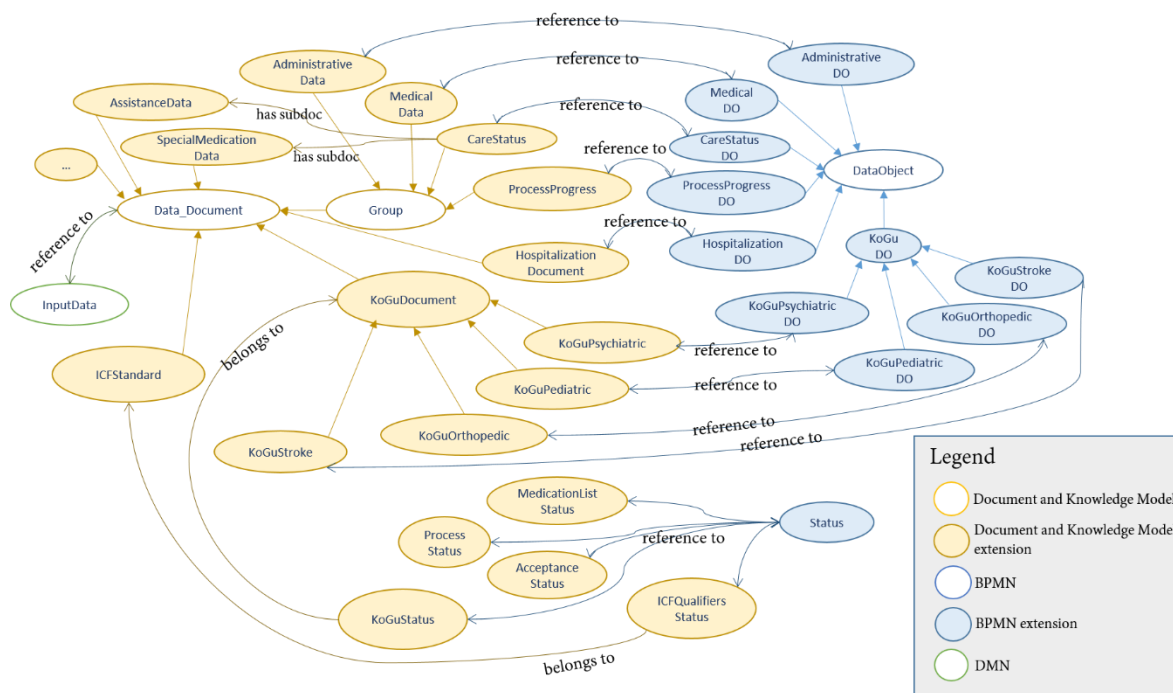


Figure 64. Extended Document and Knowledge Meta-Model and references to other meta-models

Table 17. Overview of the needed concepts from the Document and Knowledge Meta-Model

Req. Number	Concepts for the Documents and Knowledge Meta-Model
R1.2.2, R1.3.2-R1.3.5, R1.4.1, R1.4.3-R1.4.6, R2.2.2, R2.3.1-R2.3.3, R3.1.2-R3.1.5-R3.1.8	Status: Process Status, Physical Transfer Status, KoGu Status, ICF Qualifiers Status, Medication List Status, Acceptance Status; Data/Document: Patient Data, Patient Health Insurance Data, Acute Physician Data, Rehabilitation Data, KoGu, Hospitalization Document, Medical Information, ICF Standard, Medication List, Assistance Data, Process Data, Special Medication Data; Group: Medical Data, Administrative Data, Process Progress, Care Status; Relations: Belongs to, has Subdocument

4.1.4.5 Organisational Modelling View

This view deals with organisational modelling. Making the structure of the organisation explicit in a model supports the domain expert to identify quickly and easily who is performing what role in which care unit or rehab clinic.

The Organisational Meta-Model (as proposed in Emmenegger et al. (2016)) provides constructs for assigning individuals to roles, which in turn are assigned to organisation units. This structure ensures that a user can easily browse through an organisation and find a suitable person or business unit.

The organisational meta-model was extended as follows (see also Figure 65):

The role element was specialised in *Administrative Staff*, *Nurse*, *Acute Physician*, *Rehabilitation Physician*, *Patient*, *Patient Disposition* and *Transferal Manager*.

Next, the organisational unit was specialised in *Acute Hospital*, *Rehabilitation Clinic*, *Care Unit*, *Non-intensive Care Unit*, *Intensive Care Unit*, *Emergency Room*, *Site of Care* and *Health Insurance*.

The organisational meta-model was then connected with the process model as follows:

- Extended organisational unit elements are referenced with the extended pool elements from BPMN.

- Extended role elements are referenced with the extended lane elements from BPMN. For example, the transferal manger role (sub-class of role) is associated with the transferal manager (sub-class of Lane).

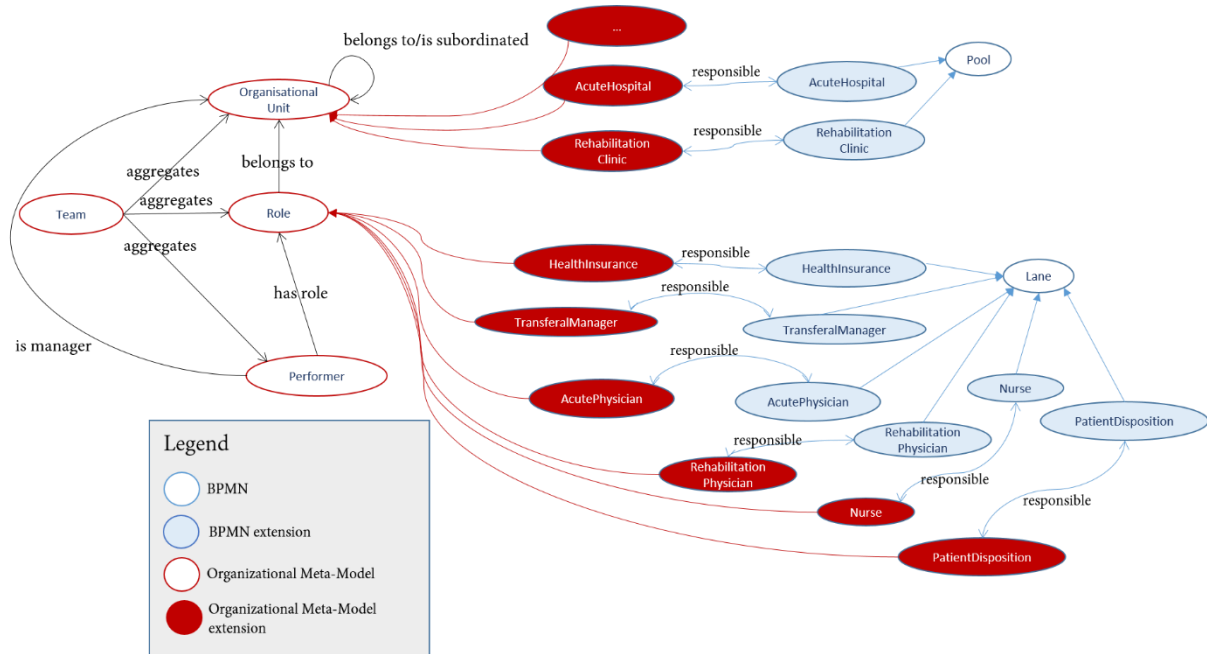


Figure 65. Extended Organisational Meta-Model and references to BPMN

Table 18 shows the concepts for the Organisational meta-model in a way that correspondent requirements are fulfilled.

Table 18 Overview of the needed concepts from the Organisational Meta-Model

Req. Number	Concepts for the Organisational Meta-Model
<i>R1.1.1-R1.1.2, R2.1.1-R2.1.2, R3.1.1</i>	Organisational Unit: Acute Hospital, Rehabilitation Clinic, Care Unit, Non-intensive Care Unit, Intensive Care Unit, Emergency Room, Site of Care and Health Insurance; Team; Performer; Role: Administrative Staff, Nurse, Acute Physician, Rehabilitation Physician, Patient Disposition, Patient, Transferal Manager; Relations: Is Subordinated; Belongs to; Is manager; Has role.

4.1.4.6 Additional semantics

According to the followed procedure, this Step aims to the add additional semantics to the meta-model in the form of constraints. Each modelling construct inherits the semantics from the correspondent modelling language. The inherited semantics is not reported as can be found in the referenced specification of each selected modelling language.

Conversely, the additional semantics (i.e, constraints and attributes) of each modelling construct are described in natural language. An excerpt of the semantics is shown in Table 19. The table shows only modelling constructs that were extended from the BPMN. The complete table containing all DSML4PTM can be found in Appendix A: Patient Transferal Management, folder A12.

Table 19. Additional semantics for BPMN modelling constructs and extensions

<i>Meta-Model</i>	Modelling construct	Additional semantics
<i>BPMN</i>	Task	It cannot be “Discretionary”.
	Manual Task PerformPhysicalAdmission, PerformTransfer, CreateTransferCaseRecord ConductFirstAssessment	It can be a “Discretionary” task.
	User Task PrepareMedicalRecordAdmission ConfirmTransfer MakeDispositionOfCase PrepareRehabAdmission PrepareKoGu ReleaseTransferCase FinalizeKoGu PrepareRehabAdmission CloseTransferCaseRecord	It cannot be “Discretionary”.
	Service Task ReleasePatientRecord, OpenPatientCase	
	Business Rule Task ApplyDefRehaCriteria ChooseRehabClinic ReapplyDefRehaCriteria	It cannot be “Discretionary”. It can have a connection to a decision of the DMN requirements diagram.
	Loop Marker Parallel Marker	It cannot be attached to a “Discretionary” task.
	Sequence Flow Exclusive Gateway Parallel Gateway	It can be connected to and from “Discretionary” tasks.
	Pool	It is linked to the “Organisational Unit” element of the Organisation Model (i.e. bridging connector).
	Pool Acute Hospital	It is linked to the “Acute Hospital Organisational Unit” element of the Organisation Model (i.e. bridging connector).
	Pool Rehabilitation Clinic	It is linked to a “Rehabilitation Clinic Organisational Unit” element of the Organisation Model (i.e. bridging connector).
	Pool Health Insurance	It is linked to a “Health Insurance Organisational Unit” element of the Organisation Model (i.e. bridging connector).
	Lane	It is linked to a “Role” element of the Organisation Model (i.e. bridging connector). It can include “Discretionary” tasks.
	Lane Transferal Manager	It is linked to a “Transferal Manager” “Role” element of the Organisation Model (i.e. bridging connector). It can include “Discretionary” tasks.
	Lane Acute Physician	It is linked to a “Acute Physician” “Role” element of the Organisation Model (i.e. bridging connector). It can include “Discretionary” tasks.
	Lane Rehabilitation Physician	It is linked to a “Rehabilitation Physician” “Role” element of the Organisation Model (i.e. bridging connector). It can include “Discretionary” tasks.
	Lane Nurse	It is linked to a “Nurse” “Role” element of the Organisation Model (i.e. bridging connector). It can include “Discretionary” tasks.
	Lane Patient Disposition	It is linked to a “Patient Disposition” “Role” element of the Organisation Model (i.e. bridging connector). It can include “Discretionary” tasks.
	Message Flow	It can be connected to “Discretionary” tasks.
	Data Object	It can be connected to either a “Group” or a “Data/Document” of the Documents and Knowledge Model (i.e. bridging connector).

KoGu Data Object	It is connected to the “KoGu” “Data/Document” of the Documents and Knowledge Model (i.e. bridging connector). It has KoGu template and KoGu Status as attributes.
Medical Data Data Object	It is connected to the “Medical Data” “Group” of the Documents and Knowledge Model (i.e. bridging connector).
Administrative Data Data Object	It is connected to the “Administrative Data” “Group” of the Documents and Knowledge Model (i.e. bridging connector).
Process Progress Data Object	It is connected to the “Process Progress” “Group” of the Documents and Knowledge Model (i.e. bridging connector).
Care Status Data Object	It is connected to the “Care Status” “Group” of the Documents and Knowledge Model (i.e. bridging connector).
Hospitalization Document Data Object	It is connected to the “Hospitalization Document” “Data/Document” of the Documents and Knowledge Model (i.e. bridging connector).
Data Association	Can also connect all new “Data Objects” with all tasks (also “Discretionary” tasks).
None Event (Standard Start) None Event (Throwing Intermediate) None Event (Standard End) Message Event (Standard Start) Message Event (Catching Intermediate) Message Event (Throwing Intermediate) Message Event (Standard End) Timer Event (Catching Intermediate)	It can be connected to “Discretionary” tasks.
Status	It is connected to six status elements of the Documents and Knowledge Model. Attributes: Data released, First assessment done, Reha conference done, Transfer date, Hospital approval, Reha approval, Patient in reha, KoGu ready, KoGu sent, KoGu accepted, KoGu rejected, Information filled in, Medication list complete, Medication list sent, Case accepted, Patient accepted. Changes progress and color based on data collection of attributes, 15 possibilities from 0 (0%) to 14 (100%) filled in (Rehab conference and KoGu rejected are not relevant for progress).

4.1.4.7 Conclusion and Problems in the Design phase

The design for the meta-model of DSML4PTM has been presented. A design procedure followed to support the creation of the meta-model. Thus, theory-based requirement *TBR4* was addressed. *TBR3* was also addressed as the choice for the selected modelling languages was made according to the elicited requirements. Since the main purpose of the language is to represent a prescribed flow, BPMN was chosen as the main language. Hence, *TBR7* was also addressed. *TBR1* is fulfilled as the language is conceived to be designed, adapted and used by the domain experts. The DSML, e.g. foresees progress status for certain indicators as well as decisions that can be automated. Finally, *TBR6* was fulfilled by providing bridging connections between concepts of different meta-models. This type of relationships (*reference to*) allow for the navigation to different views, i.e. from the main view process modelling, to one of the secondary views, i.e. decision modelling, document and knowledge modelling or organisational modelling.

During the design phase a problem was identified that prevented the fast-forward movement in the engineering lifecycle. The problem concerns the practice of updating the requirements if changes occur in the meta-model. Changes occurred because of additional aspects that were thought after the conceptualisation started. For example, this was the case with the process status, in which the requirements did not have specified how to aggregate data. The data aggregation was first conceptualised and then presented as a proposal. Therefore, in such a case, if the conceptualisation is approved, the description of the requirements might need to be updated. That is, one should go from the Design phase back to the Create phase in this case. This action reverts to the back channel between the two phases of the AMME Lifecycle. Although it is a good practice to keep the meta-model synchronised with the requirements, it is another time-consuming activity.

The problem is following formulated:

Problem 4: *The synchronization between the meta-model and requirements is a time-consuming task. When changes originated from in the meta-model, requirements were updated accordingly. Although it is a good practice (see the backward arrow from the “Design” phase back to the “Create” phase in the AMME Lifecycle), it prevents moving fast-forward to the Develop phase.*

4.1.5 Develop Phase

This sub-section describes the instantiation of the Develop Phase of the AMME Lifecycle (see the yellow light bubbles in Figure 66). In this phase, the implementation of the DSML4PTM meta-model was carried with the support of ADOxx Development Toolkit. The implementation included the creation of all the graphical notations and their association to modelling constructs.

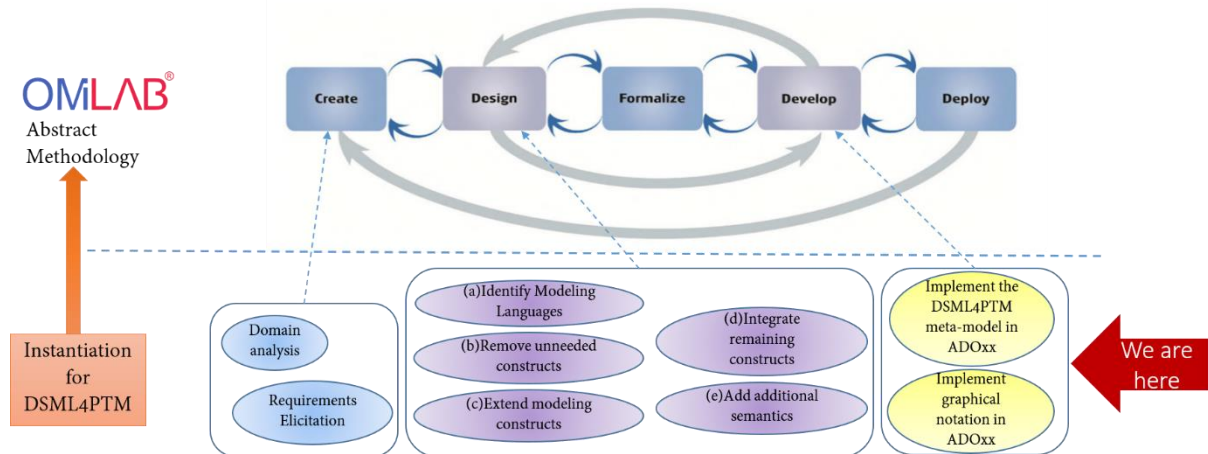




































Figure 66. Development phase instantiation for DSML4PTM

The document containing all the graphical notations of DSML4PTM can be found in Appendix A: Patient Transferal Management, folder 8. A few of them are shown in Table 20.

In result, the ADOxx library *DSML4PTM.abl* was created. The library can be found in Appendix A: Patient Transferal Management, folder 9. The library is ready to be imported in ADOxx Modelling Toolkit, in which the DSML can be used to create models. Tutorials on how to upload ADOxx libraries to create models in ADOxx Modelling Toolkit are reported in the OMiLab webpage³².

³² <https://www.adoxx.org/live/tutorial;jsessionid=C7C7389BF6CD92868BA5DA0D463947CB>

Table 20. An excerpt of the graphical notation of DSML4PTM

Role	Organisational Unit	Data Object	Document	Decision
Administrative Staff: 	Intensive Care Unit: 	General Data Object: 	General KoGu: 	General Decision 
Nurse: 	Emergency Room: 	General KoGu Data Object: 	KoGu Stroke: 	Decision Choose Rehab Clinic: 
Acute Physician: 	Site of Care: 	Medical Data Data Object: 	Patient Data: 	Decision Geriatric Rehabilitation Suitability 
Rehabilitation Physician: 	Health Insurance: 	Administrative Data Data Object: 	Acute Physician Data: 	Decision Musculoskeletal Rehabilitation Suitability: 
Patient Disposition: 	Rehabilitation Clinic: 	Process Progress Data Object: 	Patient Health Insurance Data: 	Decision Neurological Rehabilitation Suitability: 
Patient: 	Intensive Care Unit: 	Care Status Data Object: 	ICF Standard B140 S110 B730 D330B167 	Decision Inpatient Rehabilitation Suitability 
Transferal Manager: 	Acute Hospital: 	Hospitalization Document Data Object: 	Special Medication Data 	Decision Compulsory Medical Monitoring to Inpatient Rehabilitation Suitability: 

4.1.5.1 Changes from the Develop phase

The implementation of the meta-model faced some technical constraints of the adopted ADOxx Development Toolkit. These constraints led to change the design of the meta-model, thus to loop back to the Design phase. As an example, some elements that initially were conceptualised as sub-concepts (e.g. User Task, Manual Task, Service Task and Business Rule Task as sub-concepts of the class Task) had to be changed into *typeOf* concepts. In this way, the specification of an element is done while modelling (in the modelling tool) by selecting the type of elements of the selected element. For instance, after selecting a model element Task, it is possible to choose among the available types such as User Task, Manual Task, Service Task etc. (Figure 67). If such elements were left as sub-class, they had to be made visible for selection from the palette, which would overcrowd the palette. However, in terms of conceptualisation, it is more appropriate to have a sub-concept rather than a type, if its purpose is to specialise a class. Modelling elements like User Task, Manual Task, Service Task specialise the element Task (Rospocher et al., 2014).

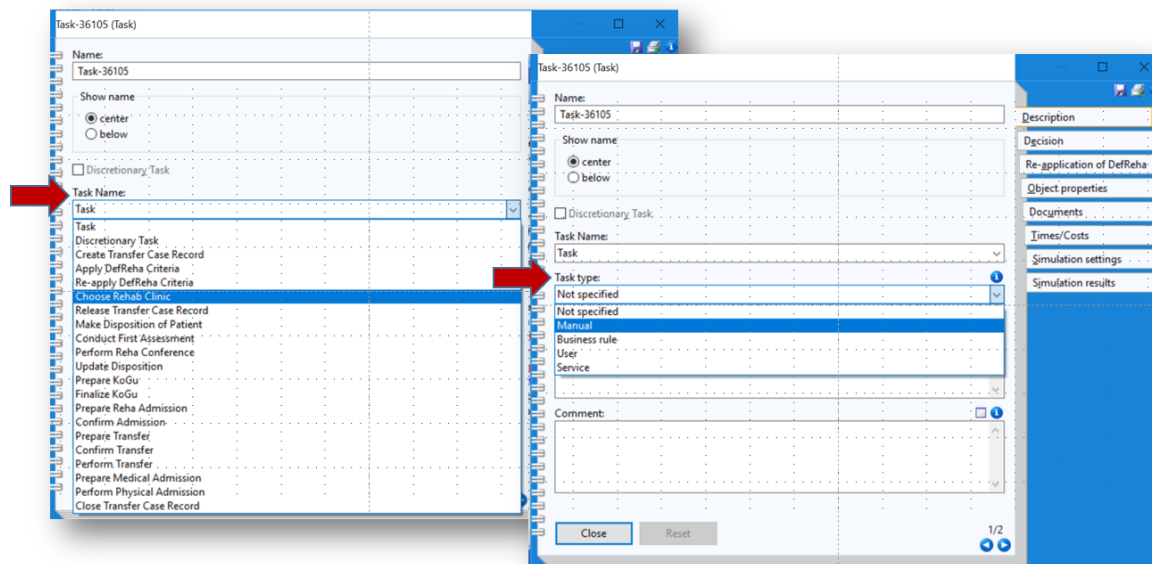


Figure 67. Result of implementation of task types and names

As Figure 68 depicts, the need for changes led to the adjustments of the meta-model (Step 1) as well as adapt the requirement specifications, accordingly (Step 2). Then, based on the new design of the meta-model, a new implementation for DSML4PTM was performed (Step 3).

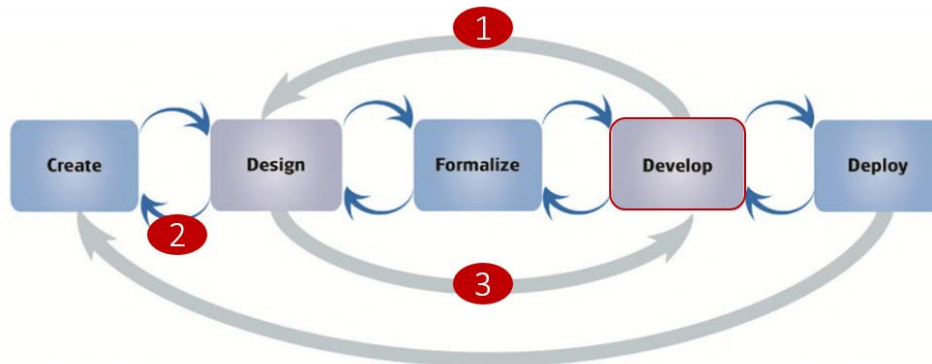


Figure 68. Propagation of changes from the Develop phase of the AMME Lifecycle

4.1.5.2 Conclusion and Considerations for the Develop phase

The changes generated during the Development phase were due to the technical constraints of the meta-modelling tool. The changes were not foreseen as, according to the AMME Lifecycle, a meta-model should be designed first. Hence, both the meta-model and the requirement specifications were adapted. Only then, the new meta-model could be re-implemented. The fourth problem hindering agility of the DSML engineering is formulated as follows:

Problem 5: *The sequential approach of first designing and then implementing a DSML is problematic due to the changes that can generate while implementing the DSML. Such changes lead back to the Design phase to conceive a new meta-model and eventually to adjust the requirement specifications. Only then, the meta-model can be re-implemented. This engineering effort prevents to quickly move on to the next phase, i.e. the Deploy/Validate phase. Moreover, the engineering effort even increases if language engineers are not experienced with a meta-modelling tool as some technical constraints can be learned only with practice.*

4.1.6 Deploy/Validate Phase

In this phase, the implemented DSML is deployed in a modelling tool as well as evaluated. The phase corresponds to the instantiation of the *Deploy/Validate* phase of the AMME Lifecycle, i.e. see bottom-right of Figure 69.

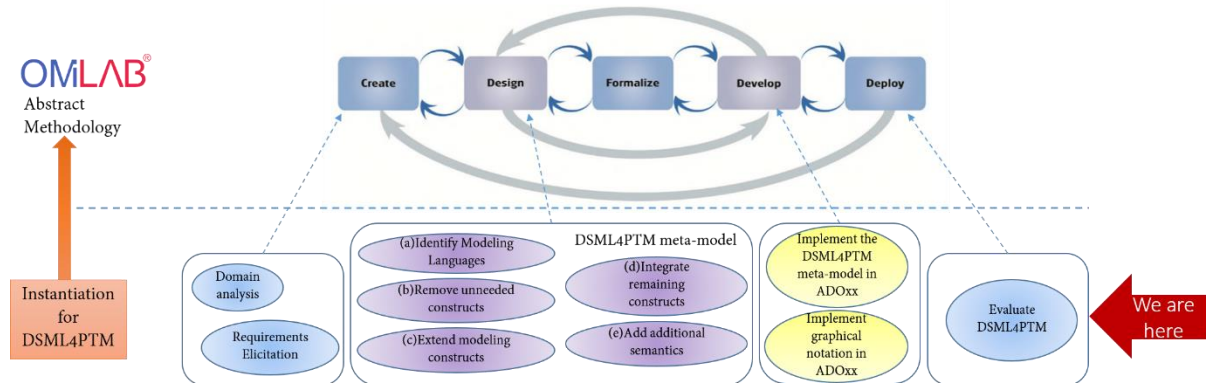


Figure 69. Deploy/Validate phase instantiation for DSML4PTM

DSML4PTM was first evaluated against the elicited application domain-based requirements (ADBRs) (see Sub-section 4.1.3.2). The evaluation was also conducted by modelling real-world scenarios with DSML4PTM (see Sub-section 0). The models in PDF format can be found in Appendix A: Patient Transferal Management, folder A10. The folder contains three sub-folders *KoGu accepted*, *KoGu rejected* and *Scenario*. In each folder there are the graphical models generated from the ADOxx Modelling Toolkit. Models in the sub-folder *KoGu accepted* represent the happy path, which focus on the elective entry case. Models in the sub-folder *KoGu rejected* also focus on the elective entry case. Finally, the third sub-folder *Scenario* contains models of the *emergency entry case - geriatric patient with stroke*. The models can also be opened in the ADOxx Modelling Toolkit. For that, both the DSML4PTM library (file with the extension “.abl”) and models (files with the extension “.adl”) can be found in Appendix A: Patient Transferal Management, folder A9.

Finally, a focus group session was performed to evaluate the perceived usefulness and cognitive effort of the language. Results of the focus group are described in Sub-section 4.1.6.4 (for documentation see Appendix A: Patient Transferal Management, folder A11).

4.1.6.1 Changes from the Deploy/Validate Phase

New requirements to amend DSML4PTM mainly originate in this phase. With the practical use of the DSML, modelling and domain experts could better understand whether the abstraction level of the language was appropriate and pitfalls were identified.

As depicted in Figure 70, new design decisions generated during the Deploy/Validate phase affected the list of requirements that were previously derived from the Create phase. Table 21 contains the changed requirements. In turn, The new requirements were addressed by designing a new version of the meta-model (Step 2 in Figure 70), which was then implemented in the ADOxx Meta-modelling Toolkit (Step 3 in Figure 70). Finally, the DSML was deployed once again (Step 4 in Figure 70).

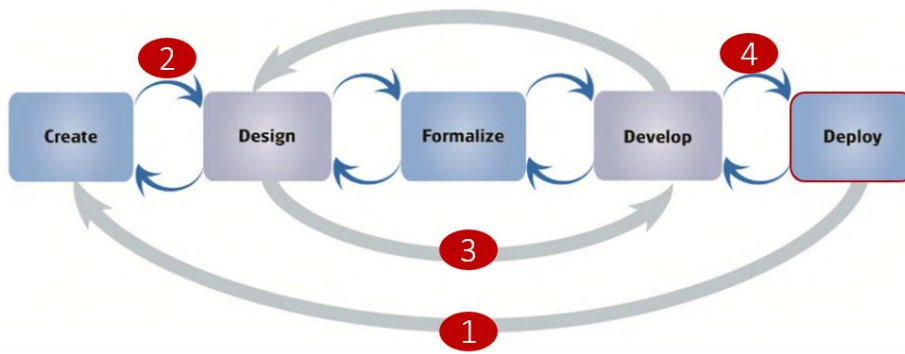


Figure 70. Propagation of changes from the Deploy/Validate phase of the AMME Lifecycle

Table 21. New Design Decisions

Number	Old Requirement	Design Decisions
<i>R1.1.1</i>	The DSML should accommodate constructs to model specific actors.	Most relevant actors will be designed with a dedicated role element in the organisation model.
<i>R1.1.5</i>	The DSML should accommodate constructs to model a system.	It is no longer relevant to show the system for the data storage. Content of the system is rather more relevant, e.g. administrative and medical data.
<i>R1.1.12</i>	The DSML should accommodate constructs to model the transfer of a patient.	This activity is modelled as standard BPMN task. No need any longer to create a dedicated element.
<i>R1.2.1</i>	The DSML should accommodate constructs to model the patient admission form.	The form can be included in the hospitalization document. In this way the need to create dedicated element will no longer be needed.
<i>R1.2.3</i>	The DSML should accommodate constructs to model the long report.	The report can be included in the administrative data. In this way the need to create dedicated element is no longer needed.
<i>R1.3.1</i>	The DSML should accommodate constructs to model the Hospital Information System (HIS) that includes the short report.	It is no longer relevant to show the system for the data storage.
<i>R1.3.5</i>	The DSML should accommodate constructs to model the Hospital Information System (HIS) that includes standard codes that classify the patient problems, i.e. Tessiner code.	It is no longer relevant to store the Tessiner code in a predefined way.
<i>R1.4.1</i>	The DSML should accommodate constructs to assign the emergency severity to a case (i.e. ESI triage).	It is no longer relevant to assign severity of a case. This is only relevant for the treatment of the patient.
<i>R2.2.1</i>	The DSML should accommodate constructs to model the patient admission form.	The form can be included in the hospitalization document. The creation of a dedicated element is no longer needed.

The following two sub-sections report the evaluation of DSML4PTM which addresses the latest list of the requirements.

4.1.6.2 Evaluation of DSML4PTM Against the Elicited Requirements

This sub-section describes the evaluation of the DSML against the derived requirements. For this, several tables were created containing the following three columns:

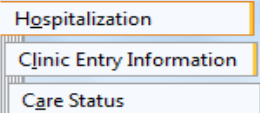
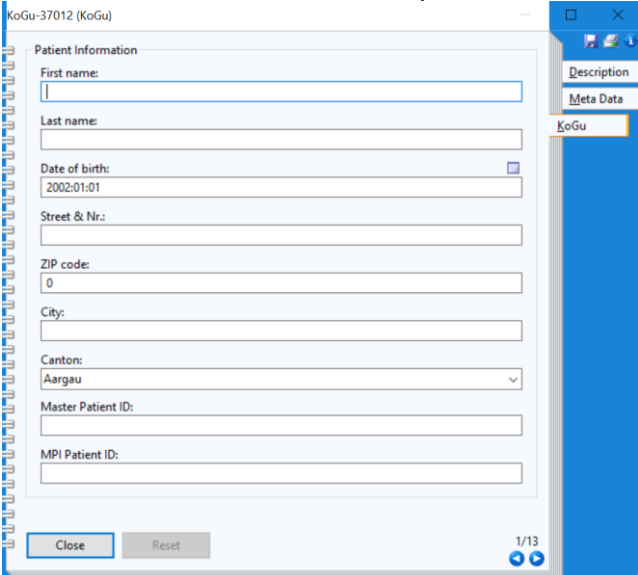

- The number of requirements,
- A small description of the requirement,
- Description about how DSML4PTM fulfils the requirement.

Each table addresses a type of requirements:

- Process requirements,
- Document requirements,
- Information systems and data requirements,
- Decisions requirements,
- Additional requirements.

Table 22 shows how document requirements were fulfilled. The rest of the tables can be found in Appendix A: Patient Transferal Management, folder A13.

Table 22. Fulfilment of document requirements

Number	Document Requirement	Fulfilment
R1.2.1	The DSML should accommodate constructs to model the patient admission form.	<p>DSML4PTM features a new data object in the modelling process view and in the document and knowledge modelling view, which is called the hospitalization document. As it is shown in the below screenshot, three notebook Sections are available to enter relevant data.</p> 
R1.2.2	The DSML should accommodate constructs to model the rehabilitation form for cost reimbursement, i.e. KoGu.	<p>DSML4PTM features a new data object in the process modelling view and document and knowledge modelling view called KoGu. Thirteen pages are available in the notebook and each Section presents dedicated relevant data.</p> 
R1.2.3	The DSML should accommodate constructs to model the long report.	<p>DSML4PTM features new documents Medical Information, Hospitalization Document and Patient Data. Each of them contains dedicated attributes.</p> 
R2.2.1	The DSML should accommodate constructs to model the patient admission form.	Fulfilled by R1.2.1.
R2.2.2	The DSML should accommodate constructs to model the rehabilitation form for cost reimbursement (KoGu).	Fulfilled by R1.2.2.

4.1.6.3 Evaluation through the Creation of Real-World Cases

The evaluation of DSML4PTM was also conducted by using the DSML to model real-world scenarios. To provide an impression of it, Figure 71 depicts an excerpt of the model representing the emergency entry case introduced in Section 4.1.3.2.4 (see model in Appendix A: Patient Transferal Management, folder A9).

Figure 71 shows how the relevant concepts and decisions that belong to different views are related to each other. The different arrows in the figure are below explained.

Arrow 1 shows the bridging connector from the pool *Acute Hospital* in the process model view to the correspondent element *Organisation Unit* in the organisational modelling view.

Arrow 2 shows the bridging connector from the data object *Hospitalization* in the process model view with the related document, which is modelled in the document and knowledge modelling view.

Arrow 3 shows the notebook with all data values of the *KoGu document*.

Arrow 4 shows the bridging connector from the sentry attached to the discretionary task *Re-apply DefReha Criteria* in the process model view, to the related sentry element in the *control element modelling view*. The latter shows the sentry with optional events (ON part) and conditions (IF part). According to the semantics inherited from CMMN, the execution of the task is possible if both *ON-part* and *IF-part* evaluate to true.

Arrow 5 shows the bridging connector from the business rule task *Apply DefReha Criteria* in the process model view to the decision construct *Decide on Rehabilitation Suitability* modelled in the Decision Requirements Diagram, i.e. decision modelling view. The latter construct takes the input *patient data* and *medical information* (both modelled in the Document and Knowledge model). Moreover, the decision construct has the knowledge source *DefReha* which refers to the external PDF containing the actual document. Additionally, rules for selecting the rehabilitation type are expressed in decision tables. Hence, all the sub-decisions that reflect rehabilitation types are modelled and used as input for the construct *Decide on Rehabilitation Suitability*. For example, the neurological rehabilitation type has three interfaces: *Inpatient Rehabilitation*, *Compulsory Medical Monitoring* and *Medical Monitoring to Inpatient Rehabilitation*. The three interfaces are modelled as further decision constructs that in the meta-model relate to *Neurological Rehabilitation Suitability*.

Arrow 6 leads the process status element in the *process model view* to the notebook containing all the attributes for which the status is determined, i.e. *data released*, *first assessment done*, *Rehab conference done*, *transfer date*, *KoGu ready*, and so on.

Noteworthy is the integration of CMMN and BPMN in the process model view, which allows specifying the actor who would perform a discretionary task by placing the task on the appropriate lane. This specification is not foreseen in the current version of CMMN 1.1 (OMG, 2016a).

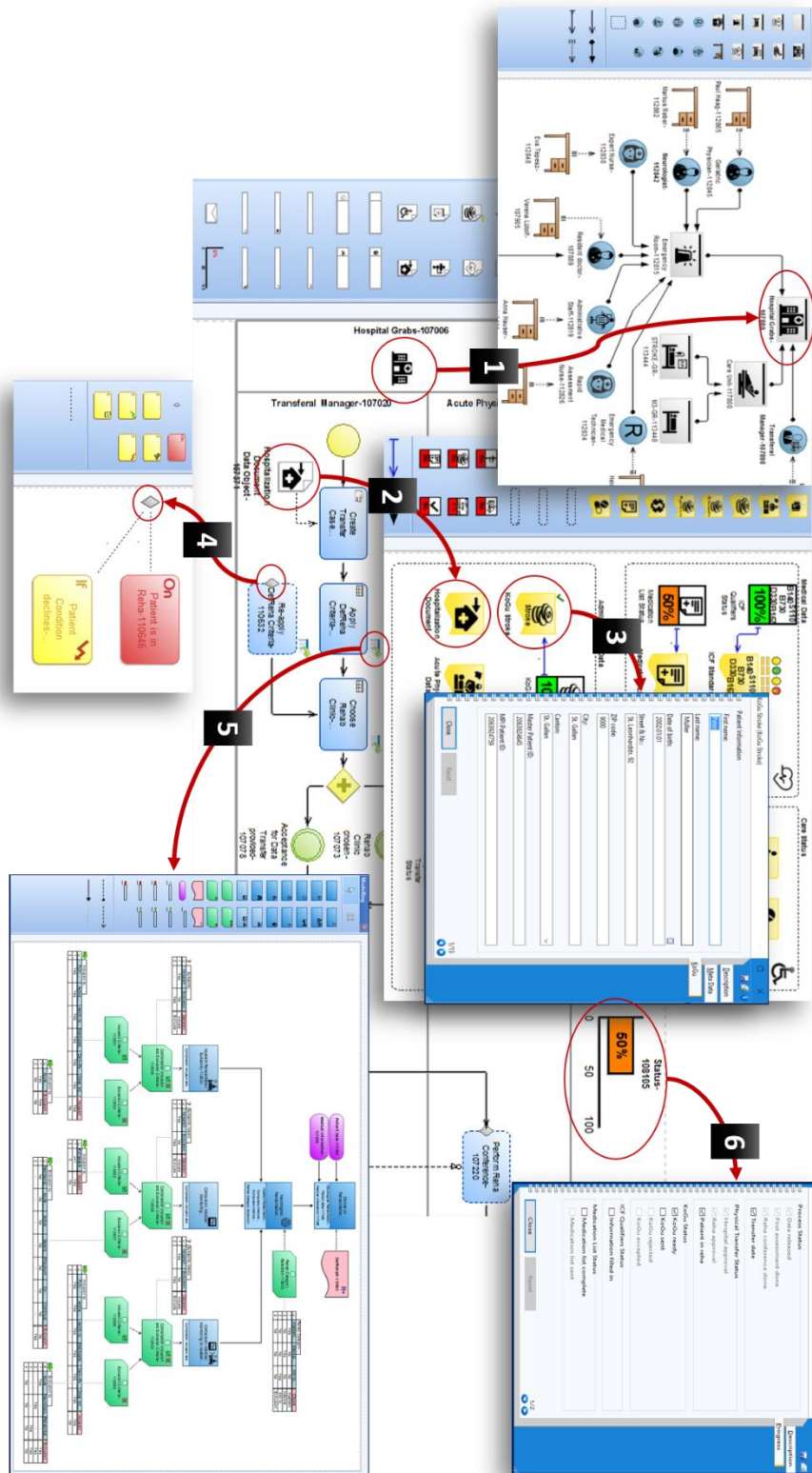


Figure 71. Part of the reference model implemented in the ADOxx Development Toolkit

4.1.6.4 Focus Group Evaluation

Models created in the previous evaluation were used in a two-hour focus group session with four modelling experts and one domain expert for an evaluation on the *perceived usefulness* of the language and its *cognitive effort*. In the first hour, the new modelling language was introduced to the participants and then a walk-through session explained how the new language can be applied. In the following hour, the participants were given a hands-on session and they were asked to extend the reference process with two new scenarios:

1. Discharging criteria complied with the standard DefReha should be re-applied in case the cost reimbursement form (KoGu) is rejected.
2. The cost reimbursement form (KoGu) needs to be revised as some information is missing.

Finally, a questionnaire was provided to perform a qualitative evaluation.

The questionnaire was developed based on the guidelines proposed in (Tullis & Albert, 2013). It was conducted on an individual basis and within a time frame of at least half an hour. The questionnaire contained five sections, i.e. four questions on general background; eleven questions on modelling background; three open questions on the perceived usefulness of the new modelling language; five open questions on the cognitive effort of the new modelling language; and three open questions as a feedback for future improvements.

In the following, we summarize the outcome of the questionnaire with respect to the perceived usefulness and cognitive effort.

Perceived usefulness: All participants agreed on the high potential of the DSML in fostering communication and collaboration among domain experts. Modelling experts stated that the new language simplifies the modelling process, which helps improve model quality. They also emphasised that the language can be seen as a basis for the integration of different information systems (e.g. health information systems, patient administration systems, health records) and automated verification.

Cognitive effort: All participants agreed on the ease-of-use of the new language. For which a metric is the amount of time that is required to learn the usage of a language. All the participants stated that the usage of the language could be learned within a short period of time. This was backed by the fact that participants could propose meaningful models for the two new scenarios within less than half an hour. Of course, to apply the language in real settings would need more time to get acquainted with it. Further metrics we looked at are the appropriate abstraction of language elements and their default values, the graphical notation of the concrete syntax, and the support at design time in creating meaningful models. The latter was mentioned as one of the main strengths as during the hands-on session, participants perceived they could not connect model elements with each other randomly. In line with this, modelling experts mentioned the reduction of modelling mistakes as a strength of the new language. One agreed-upon drawback of the language is the difficulty to understand when to use a sentry rather than a gateway. We noted that during the hands-on session, those with a strong BPMN background tended to use gateways only. One suggestion was to encourage the usage of sentries as a best practice while minimising the use of gateways.

4.1.6.5 Conclusion and Problems in the Deploy/Validate Phase

In this phase, DSML4PTM was subject to three types of evaluation: (1) against the application domain-based requirements, (2) by modelling rea-world scenarios and (3) by conducting a focus group with modelling and domain experts. The Deploy/Validate phase generated the highest amount of changes to accommodate in DSML4PTM. The problem is thereby formulated as follows:

***Problem 6:** In order to integrate new requirements in the DSML, the engineering lifecycle is re-iterated from the initial phase (i.e. Create phase). For each iteration, all the engineering phases (Create, Design, Develop and Deploy/Validate) are sequentially re-performed until a new version of the DSML is ready to be used. Going through each phase is an engineering effort as well as time-consuming, thus it prevents the quick versioning of a DSML.*

4.2 Case 2: Business Process as a Service (BPaaS)

During a project CloudSocket³³ (Woitsch & Utz, 2015) it turned out that existing modelling languages were not expressive enough for modelling the needs in the application domain business process as a service (BPaaS). Like the first case, the work in this application domain gave the chance to analyse the agility issue from a practical perspective.

Differently from DSML4PTM, the BPaaS DSML focuses on both the human and machine interpretation of knowledge.

This case study, like the Patient Transferal Management case study, follows the case design described in the introduction of Chapter 4. The remainder of this section is structured as follows: first, the introduction and motivation of the work are provided. Then, the related work is reported to provide the theoretical framework for the DSML. Next, the instantiations of the AMME Lifecycle are introduced and then each phase is summarized along with faced problems. Differently from DSML4PTM, the BPaaS DSML engineering lifecycle (*Create, Design, Develop and Deploy/Validate*) was iterated three times. The iterations were due to the arise of new requirements after the language evaluation. The three engineering iterations are described from Sub-section 4.2.6.3 to 4.2.6.5 and the new requirements are reported after the first and second iterations.

4.2.1 Introduction and Motivation

It is a challenge for today's enterprises to continuously align business and IT in a rapidly changing environment. According to Gartner (2014) enterprises are facing a new era of enterprise IT, which is “characterized by deep innovation beyond process optimization, exploitation of a broader universe of digital technology and information, more-integrated business and IT innovation, and a need for much faster and more agile capability.” To gain appropriate benefit from the digitalization business and IT should be deeply integrated.

BPaaS aims to bring end-to-end business processes in the cloud. In this context, the alignment between business and IT had the purpose of supporting the retrieval of suitable cloud offerings for the specified business requirements. Cloud offerings are typically specified in IT language. In contrast, requirements come from the business side, thus are specified in business language. To support the mapping between the business requirements and the IT specifications a component was developed. Such components require the modelling of both business requirements and service specification in an enterprise ontology. The latter allows automating the Business-IT alignment.

However, the specification of an enterprise ontology is difficult for both business people and cloud service providers. Therefore, the task was to create a DSML that allows to easily model both business requirements and service specifications.

³³ <https://site.cloudsocket.eu/>

4.2.2 Related Work

BPaaS represents an initial field of research. Most of the research work proposed focuses on how to define BPaaS and the respective candidate architectures to realize it (Amziani, Melliti, & Tata, 2012). Some work has concentrated on dealing with security aspects, e.g. anonymization-based protocols for BPaaS fragments (Azzini et al., 2013). Initial work has been conducted on how elasticity can be realised for BPaaS through a specific formal model and a respective elasticity framework (Lynn et al., 2014).

Modelling the business processes, workflows and services is part of Enterprise Modelling - the description and definition of the processes, structure, information and resources of an enterprise. According to Fox and Gruninger (1998) an enterprise model must supply the information and knowledge necessary to support the operations of the enterprise. Enterprise modelling techniques are developed in several fields such as business process modelling, information modelling, systems modelling, and enterprise architecture.

As already seen in the literature review chapter, OMG has developed several specialised modelling languages for enterprise modelling, for example Business Process Model and Notation (BPMN), Case Management Model and Notation (CMMN), the Decision Model and Notation (DMN). The primary purpose of these graphical modelling languages is to support communication between human stakeholders. Adding formal semantics to business processes enables machine reasoning and allows exploiting the full potential of process models. This semantic lifting can be achieved by representing a model with ontologies. De Nicola et al. (2008) already mentioned the use of semantic lifting for the alignment of business and IT. Other applications for semantic lifting are process automation (Hepp et al., 2005), process mining (Azzini et al., 2013) or learning (Emmenegger et al., 2017).

The purpose of ontologies in enterprise modelling is to formalize and establish the sharing, reuse, assimilation and dissemination of information across all organisations and departments within an enterprise. Developing enterprise ontologies started in the 1990s with TOVE (Fox, 1992), The Edinburgh Enterprise Ontology (Uschold et al., 1998) and the organisational memory (Abecker et al., 1998).

More recent work is the Context Based Enterprise Ontology (Leppänen, 2007). Den Haan (Den Haan, 2009) has used an enterprise ontology to realize a Model-Driven Enterprise Engineering.

In this research an enterprise ontology was used which is based on the concepts of the ArchiMate Standard (The Open Group, 2012a) and extend it with concepts for BPaaS. ArchiMate is an integrated modelling language for enterprise architectures (EA). It is consistent with the TOGAF (The Open Group, 2011) framework. The overall enterprise architecture comprises a set of closely inter-related architectures: Business Architecture, Information Systems Architecture, and Technology Architecture. Another well-known EA framework is the Zachman Framework, a two dimensional matrix, in which the cells contain models (Zachman, 2008).

Hinkelmann et al. (2016) combined Enterprise Architecture modelling and Enterprise Ontologies for continuous alignment of business and IT. They show the potential of having both graphical and ontological representations in one environment. The work in this case study builds on that approach, where ontologies are eventually used to support the design of graphical models.

4.2.3 Methodology

The development of the BPaaS DSML is supported by the AMME (or OMiLab) Lifecycle (Karagiannis, 2015). Differently from DSML4PTM, which is limited to a human-interpretable representation, BPaaS DSML aims for both the human and machine interpretation of knowledge. The human-interpretable knowledge is enabled by graphical models while an ontology allows for the machine interpretation of the models. In order to develop BPaaS DSML the AMME Lifecycle was instantiated twice:

- Instantiation for the human interpretation of BPaaS DSML,
- Instantiation for the machine interpretation of BPaaS DSML.

Both instantiations contributed to conceiving the general architecture of the BPaaS Design Environment, which is described in Sub-section 4.2.4.

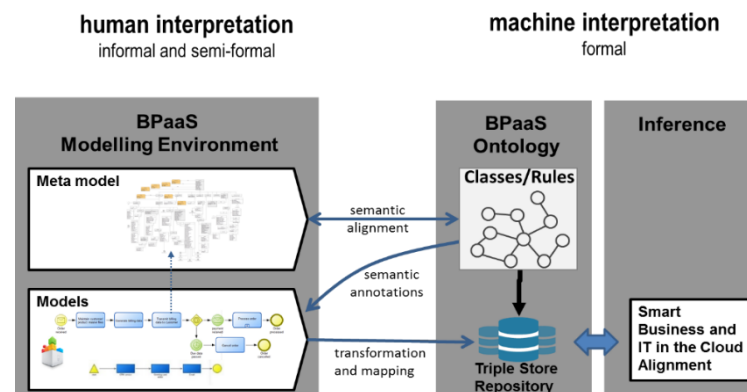
Sub-section 4.2.6 summarizes each engineering phase of both instantiations. Problems identified during the engineering lifecycle are reported at the end of each phase. This is done by providing a comparison with the six problems already identified during the development of DSML4PTM.

4.2.4 BPaaS Design Environment Architecture

The BPaaS DSML was deployed in a hybrid ADOxx-based modelling tool (i.e. the BPaaS Design Environment – see Figure 72), which foresees the design of domain-specific business processes and related requirements as well as the design of workflows and related cloud specifications. As afore-mentioned, the BPaaS Design Environment was conceived by instantiating twice the AMME Lifecycle, once for the human interpretation of BPaaS DSML and once for its machine interpretation.

The first one encompasses the BPaaS meta-model for the human-interpretable, graphical modelling languages. The graphical models can be semantically annotated with the ontological concepts, which are defined in the BPaaS Ontology. An inference engine uses the BPaaS Ontology and semantic rules to enable the smart IT-Business alignment in the Cloud.

Both BPaaS Ontology and meta-model development have been synchronised as the ontology contains class definitions describing the intended semantics of the elements of the graphical modelling language. This approach provides the possibility of modelling both business processes and workflows, and annotating elements modelled with corresponding functional and non-functional requirements and specifications, respectively.



4.2.5 Create, Design, Formalize, Develop and Deploy/Validate Phases

Figure 73 depicts the concrete steps (see bubbles) of each phase of the two AMME Lifecycle instantiations. Bubbles with the same colour mean they belong to the same phase. Dashed arrows also indicate the reference to a phase. The information flow among steps is shown through solid blue arrows.

Due to changes in requirements derived by feedback from business process managers, cloud computing experts and broker from the project team, both instantiations were iterated three times during the project. Each iteration led to building three new prototypes, i.e. see the three yellow bubbles that belong to the instantiation for machine interpretation in Figure 73: Semantic Lifting Prototype (*first iteration*), Matchmaking Prototype (*second iteration*), and Questionnaire Prototype (*third iteration*). Most of the steps in the bubbles with the same colour were run in parallel. From Sub-sections 4.2.5.1 to 4.2.5.8 each phase is elaborated including the faced problems.

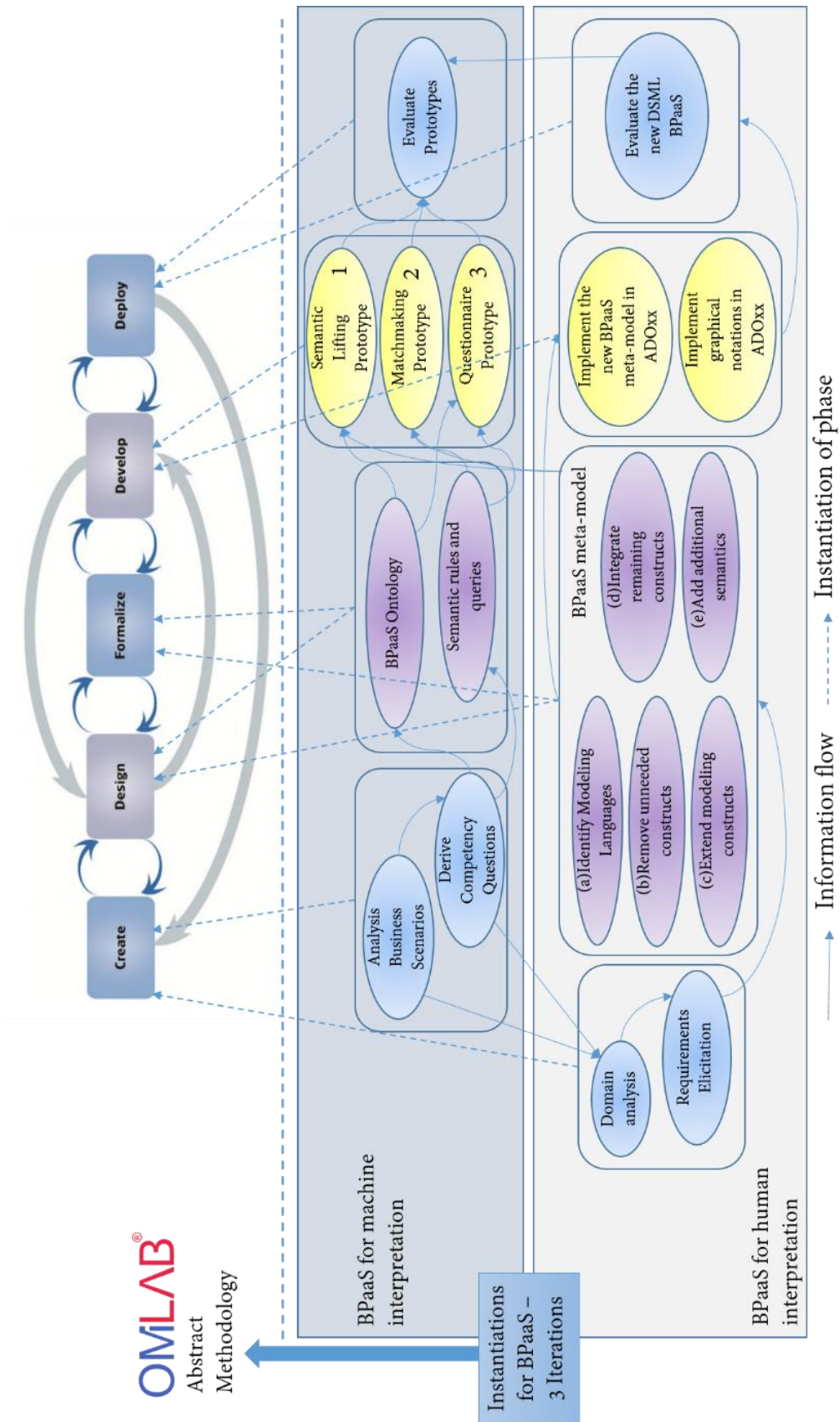


Figure 73. Two AMME Lifecycle instantiations for the human and machine interpretation of BPaaSDSML

4.2.5.1 Create Phase

In this phase, the analysis of business scenarios served as a starting point, since they represent real situations as they occur in enterprises. The business scenarios were developed in workshops conducted with project members of the European project CloudSocket, which included business process managers, cloud computing experts and cloud brokers. Each of the three iterations corresponds to the creation of a new business scenario. The three scenarios were the following (from the oldest to the newest):

- the Sending Christmas Card Process (*the first iteration* – see (Woitsch et al., 2016), for documentation see in Appendix B: Business Process as a Service, folder B1);
- the Social Media Campaign Process (*the second iteration* - see (Hinkelmann et al., 2016), for documentation see in Appendix B: Business Process as a Service, folder B2);
- the Send Invoice Process, (*the third iteration* - see (Kritikos et al. 2018)).

The first scenario mainly contributed to “derive competency questions”, which served to determine the scope of the ontology. The following two scenarios contributed cumulatively to the competency questions, which were implemented directly as semantic rules and queries through SPARQL. Thus, the scope of the ontology evolved over the iterations. Competency questions are introduced by Gruninger and Fox (1995) as a method for enterprise engineering and ontology scope determination (Uschold & Gruninger, 1996). This approach is widely known and was amongst others adopted in (De Bruijn, 2003; De Leenheer & Mens, 2008; Cardoso, 2010). One example of a competency question is as follows: *Which parts of my business process can be served by existing workflows?* (the list of the first competency questions can be found in the documentation in Appendix B: Business Process as a Service, folder B1).

- *First iteration.* The domain and scope determination correspond to steps 1 to 5 of the approach for ontology development (Noy & McGuinness, 2001). Moreover, the literature review on existing ontologies was performed in parallel to ensure the coverage of existing material. This was mainly an effort during the first iteration (for documentation see in Appendix B: Business Process as a Service, folder B1). In the second and third iteration, ontologies were adapted and extended.
- *Second iteration.* The BPMN convention to name activities by a verb and a noun (Silver, 2011); the analysis of existing standards such as APQC Process Classification Framework (APQC 2014) and Cloud Service Level Agreement Standardisation Guidelines (EC Cloud Select Industry Group (C-SIG), 2014) resulted in an adaption of the functional requirement and specification.
- *Third iteration.* The analysis of 46 recent scientific papers (i.e. from 2009 to 2018) (Giovanoli, 2019) of cloud services as well as the analysis of four existing cloud marketplaces, i.e. Ymens³⁴, IBM³⁵, Also³⁶ and UK digital marketplace³⁷ led to adaptations of the descriptions of non-functional requirements.

Results of each iteration were used as input for the domain analysis of BPaaS DSML. The results contributed to set the requirements for the design of the meta-model (see information flow arrows in Figure 73). This phase was subject to continuous adaptation and feedback through typical collaboration instruments used within the CloudSocket consortia, i.e. physical

³⁴ <http://www.ymens.ro/en/frontpage>

³⁵ www.bluemix.net

³⁶ www.alsocloud.ch

³⁷ <https://www.digitalmarketplace.service.gov.uk/g-cloud>

meetings, internet workshops, presentations and collaborative development. Thus, further iterations occurred within each of the three iterations.

4.2.5.2 Problems in the Create phase and comparison with DSML4PTM

The problems identified in the Create phase of DSML4PTM are compared with the experience in BPaaS. Table 23 shows the comparison. No further problems were identified in this phase.

Table 23. Comparison of problems identified in the Create phase for DSML4PTM and BPaaS

<i>Problems in DSML4PTM</i>	<i>Problems in BPaaS</i>
Problem 1. <i>Language engineers and domain experts have different types of expertise. This might result in misinterpretation of requirements.</i>	The issue also manifested in the creation phase of BPaaS DSML. The domain expertise (i.e. specifying Cloud requirements in business-like language) had to be deepened until a stable list of requirements was delivered. Meetings and workshops with domain experts helped the language engineers to gradually increase understanding.
Problem 2. <i>The extraction, documentation, prioritization, and categorization of requirements is a time-consuming manual task.</i>	The time-consuming task includes extracting, documenting prioritizing and categorizing manifested with the manual job of creating competency questions. The latter, nevertheless, helped to extract relevant aspects for the language concisely. The sequential approach in this case reverts to the transformation of competency questions to ontology concepts and semantic rules (i.e. from creation phase to design phase), where the ontology was used as a conceptual meta-model of the DSML.
Problem 3. <i>The update and synchronization of requirements is a time-consuming task.</i>	The time-consuming task of updating and synchronizing among different requirements did not emerge as for each engineering iteration there was only the literature review and one business scenario as sources.

4.2.5.3 Design Phase and Formalisation Phase

The design and formalisation phases were done in parallel. Results from the previous phase were considered to design both the meta-model and BPaaS Ontology (see information flow arrows in Figure 73). The meta-model and the ontology were designed in parallel too so to ensure consistency. Namely:

1. In the instantiation for the *machine interpretation of BPaaS DSML*, the BPaaS Ontology was first conceptualised as an extension of the ArchiMEO³⁸ enterprise ontology and then formalized into the RDF(S) ontology language. Semantic rules and queries were also created and formalized in SPARQL. The ontology editor TopBraid³⁹ was adopted (for the documentation of BPaaS Ontology⁴⁰ see in Appendix B: Business Process as a Service, folder B1).
2. In the instantiation for the *human interpretation of BPaaS DSML*, the steps from (a) to (e) introduced in Sub-section 4.1.4, were performed to design the meta-model. The conceptualisation and formalisation of the meta-model can be found in Appendix B: Business Process as a Service, folder B1. The formal meta-model is specified in first-order logic-based formalism FDMM (Fill et al., 2012).

³⁸ <http://ikm-group.ch/archimeo>

³⁹ <https://www.topquadrant.com/tools/ide-topbraid-composer-maestro-edition/>

⁴⁰ The full set of BPaaS Ontologies is retrievable at: <https://github.com/BPaaSModelling/CloudSocket-Ontology>

4.2.5.4 Problems in the Design and Formalisation Phase and Comparison with DSML4PTM

In the Design phase of BPaaS DSML, it was identified a similar as problem 4 in DSML4PTM. Table 24 shows the description of the problem in relation to problem 4. Below, a new problem is presented, which did not arise in DSML4PTM.

Table 24. Comparison of problems identified in the Design phase for DSML4PTM and BPaaS DSML

<i>Problem in DSML4PTM</i>	Problems in BPaaS DSML
<i>Problem 4. The synchronization between the meta-model and requirements is a time-consuming task.</i>	A similar problem was faced in BPaaS DSML. When a change occurred in the ontology the related competency questions was aligned to avoid inconsistencies.

The new problem identified in the phase concerns the Formalize phase, which was not performed for DSML4PTM. This is a quite a heavy engineering effort, which requires high expertise in both conceptual models as well as logic-based formalism. If changes are accommodated in the language, each iteration leads to adapt the formal description of the meta-model, resulting in a time-consuming task (for documentation see Appendix B: Business Process as a Service, folder B1). The problem is formulated as follows:

Problem 7: *The formalisation of the meta-model is an intensive and time-consuming engineering task, which requires high expertise. According to the followed methodology, this activity is in between the design and the development steps. That is, every new engineering iteration that changes the meta-model leads to change the correspondent formal representation to keep consistency. This problem prevents the quick release of new versions of a DSML.*

4.2.5.5 Develop Phase

In this phase, the steps of the two instantiations (human and machine interpretation of BPaaS DSML) were performed in parallel (see the yellow bubbles in the initially introduced Figure 73). In each iteration a new prototype was developed. BPaaS DSML was iteratively adapted and used first in the Semantic Lifting prototype and then in the Matchmaking prototype. The three engineering iterations are elaborated below.

1. In the first iteration, the BPaaS Ontology (including semantic rules and queries) and BPaaS DSML meta-model were the basis to implement the Semantic Lifting Prototype (see correspondent information flow arrow in the above Figure 73). Namely, the latter was built such that classes and instances from the BPaaS Ontology could be selected for the semantic annotation. Thus, the prototype⁴¹ allows performing the semantic annotation of models created with the from the ADOxx modelling toolkit (for documentation see Appendix B: Business Process as a Service, folder B2). The BPaaS DSML for human interpretation was implemented in the ADOxx Development Toolkit. The implementation included the creation of graphical notations for BPaaS DSML. The resulting DSML BPaaS was then ready to be used in the ADOxx Modelling Toolkit. In the first iteration ADOxx libraries were developed around the scenario Christmas Card process (the

⁴¹ The prototype can be downloaded from the GitHub repository: <https://github.com/BPaaSModelling/BPaaS-Annotation-WebService>

libraries for the meta-model and model can be found in Appendix B: Business Process as a Service, folder B3).

2. In the second iteration, the new version of BPaaS Ontology along with semantic rules and queries were used as basis for implementing the Matchmaking prototype. The latter enable the smart Business-IT alignment in the cloud such that the suitable workflows and cloud services are retrieved to serve parts of a business process (the documentation can be found in Appendix B: Business Process as a Service, folder B2). Thus, transformations from models to ontologies should occur and the result is then used as input for the prototype⁴². Similarly, the meta-model of BPaaS DSML was also adapted, extended and implemented in the ADOxx Development Toolkit. The first application scenario that underpinned the iteration was the Social Media Campaign process. Further on, BPaaS DSML was refined to model the Send Invoice scenario (for documentation see Appendix B: Business Process as a Service, folder B3).
3. In the third iteration, similarly to the second iteration, a new version of BPaaS Ontology, along with semantic rules and queries were used as basis for implementing the Questionnaire prototype. The Send Invoice process underpinned this iteration (the process can be found in Appendix B: Business Process as a Service, folder B3). The Questionnaire prototype is stand-alone. It was later embedded in the BPaaS Design Environment so to be triggered from any process fragments within the design environment. Moreover, the prototype foresees an ontology-based meta-model (Hinkelmann et al., 2018). A new version of BPaaS DSML meta-model was, therefore, not required. Specifically, the Questionnaire prototype consists of two components: (1) the web app⁴³ that allows for the user interaction with the questionnaire, and (2) the web service⁴⁴ that allows for the propagation of the ontologies to the web app so that the questionnaire is graphically displayed.

4.2.5.6 Problems in the Develop Phase and Comparison with DSML4PTM

The problem faced in the Develop phase for BPaaS DSML was the same as in DSM4PTM. The problem is described in Table 25. No further problems were identified.

Table 25. Comparison of the problem identified in the Develop phase for DSML4PTM and BPaaS

<i>Problem in DSML4PTM</i>	Problems in BPaaS DSML
<i>Problem 5. The sequential approach of first designing and then implementing a DSML is problematic due to the changes that can generate while implementing the DSML</i>	The same problem partially raised for BPaaS. On the one hand the tool constraint issue did not raise as most of the new concepts were entered in the form of attributes. Thus the abstraction level of concepts was not an issue. On the other hand, it applies the same sequential issue where for each new requirement, its implementation (i.e. in the development phase) was only done after its conceptualisation (i.e. in the design phase) in the meta-model.

⁴² The matchmaking prototype is retrievable at: <https://github.com/BPaaSModelling/BPaaS-Annotation-Matcher>

⁴³ https://github.com/BPaaSModelling/Questionnaire_v3-WebApp

⁴⁴ https://github.com/BPaaSModelling/Questionnaire_v3-WebService

4.2.5.7 Deploy/Validate Phase

Evaluation activities were performed at the end of each engineering iteration. For this, the scenarios were implemented in the prototypes and the results were shown to project partners so to collect feedback. Namely:

- In the first iteration the Semantic Lifting approach and first version of the BPaaS DSML were evaluated by (1) creating models for the two scenarios Sending Christmas Card and Social Media Campaign through the Semantic Lifting prototype, (2) showing demos to the project partners.
- In the second iteration, the Matchmaking approach and new version of the BPaaS DSML were evaluated by (1) creating models for the Social Media Campaign and the Send Invoice scenarios through the Matchmaking prototype, (2) showing demos to the project partners.
- Finally, in the third iteration, the Questionnaire approach and ontology-based meta-model were evaluated by (1) executing the Send Invoice scenario in the Questionnaire prototype and (2) showing demos to the project partners.

Figure 69 shows the information flow from the evaluation of the DSML to the evaluation of the approaches. Such flow indicates that the new versions of BPaaS DSML were used in the creation of models within the considered scenarios (the models implemented in ADOxx can be found in Appendix B: Business Process as a Service, folder B3).

4.2.5.8 Problems in the Deploy/Validate Phase and Comparison with DSML4PTM

Table 26 contains the problem faced in this phase, which is similar to the already introduced problem 6.

Table 26. Comparison of the problem identified in the Deploy/Validate phase for DSML4PTM and BPaaS

<i>Problem in DSML4PTM</i>	Problems in BPaaS DSML
<i>Problem 6.</i> <i>In order to integrate new requirements in a DSML, the engineering lifecycle has to be re-iterated from the initial phase (i.e. Create phase).</i>	The same problem was faced in BPaaS. Significant changes raised at the end of the first two iterations. Time-wise, each iteration was equivalent to approximately six to eight working months.

4.2.6 Engineering Lifecycle Iterations for BPaaS DSML

This sub-section presents the relevant aspects that led to iterate the engineering lifecycle three times. First, the application scenario for the domain analysis is presented. Then, the initial version of the meta-model is introduced. Next, each of the engineering lifecycle iteration is briefly described, along with the new requirements that arose before each iteration.

4.2.6.1 Application Scenarios for the Domain Analysis

The domain analysis in the Create phase led to the identification of a set of relevant business processes. In this sub-section, only the most recent business process is presented “Send Invoice”. This business process was identified as one of the most widespread processes in industry.

Figure 74 shows the business process, which starts with the “Manage Customer Relationship” activity. Next, an exclusive gateway splits the business process flow between either creating a new invoice or updating an existing one. Then, invoice completeness is checked, and finally the invoice is sent.

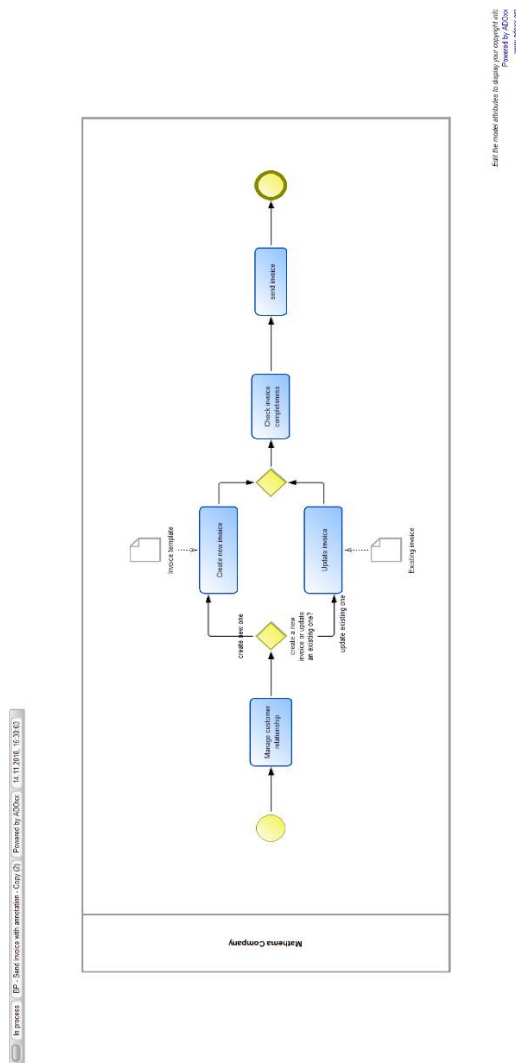


Figure 74. Send Invoice Business Process

A potential workflow model that implements the business process Send Invoice is depicted in Figure 75⁴⁵. Workflow activities present a deeper granularity than activities in business processes, and typically leads to a higher number of activities. Such activities are bounded with a cloud service. Thus, a workflow is meant to be executable and subsequently deployable in the cloud. BPMN lanes of workflows represent IT systems or cloud services. This means, all the activities in a workflow lane are supported by the IT system placed in the lane.

The workflow shown in Figure 75 incorporates the following cloud services:

1. Customer Relationship Management system (CRM) like YMENS⁴⁶ (see top lane of the workflow);
2. An invoicing system that includes a document generator like InvoiceNinja⁴⁷ (see middle lane of the workflow);
3. An e-mail and file system supported by Invoice Ninja (see bottom lane of the workflow). Customer data are stored in the CRM, and the invoice is transmitted via an email system.

As afore-mentioned, the business processes and workflows were used to create competency questions from which the BPaaS Ontology was derived. The BPaaS Ontology supported the Design phase of the BPaaS DSML. The relevant aspects of the Design phase together with the Develop and Deploy/Validate phases are described in the following sub-section.

⁴⁵ A higher resolution of the workflow model can be found in Appendix B: Business Process as a Service, folder B5

⁴⁶ <http://www.ymens.ro/en/frontpage>

⁴⁷ <https://www.invoiceninja.com/>

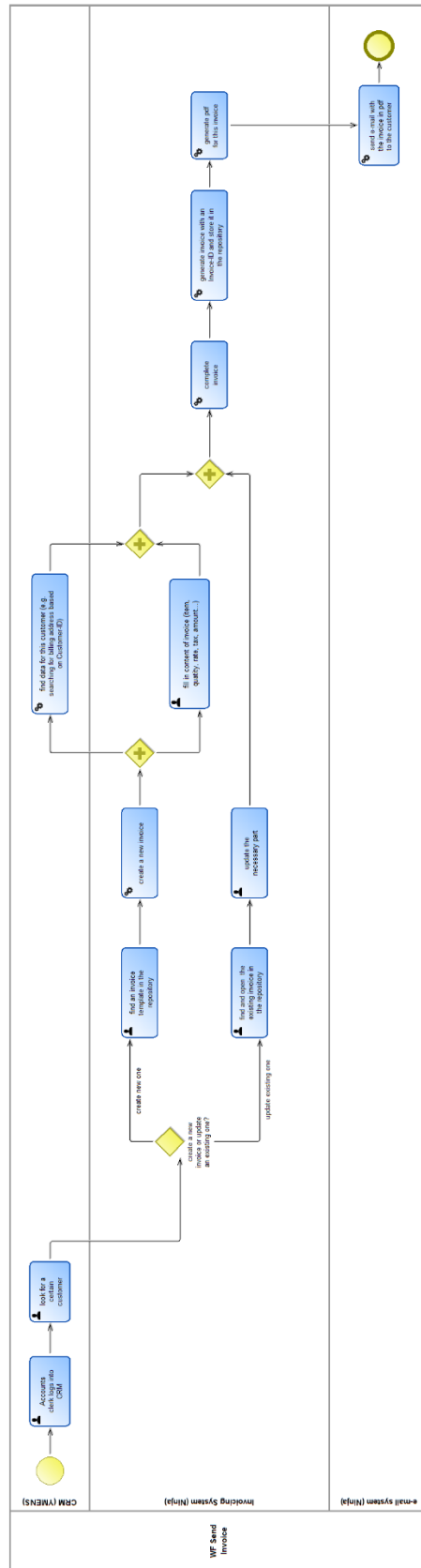


Figure 75. A workflow implementing the business process Send Invoice

4.2.6.2 Initial Version of BPaaS DSML

During the Design phase of the first engineering lifecycle, the BPaaS meta-model was defined. Such definition includes (a) domain-specific business layer and (b) the IT-Cloud relevant technical layer as well as the interaction between them (see Figure 76).

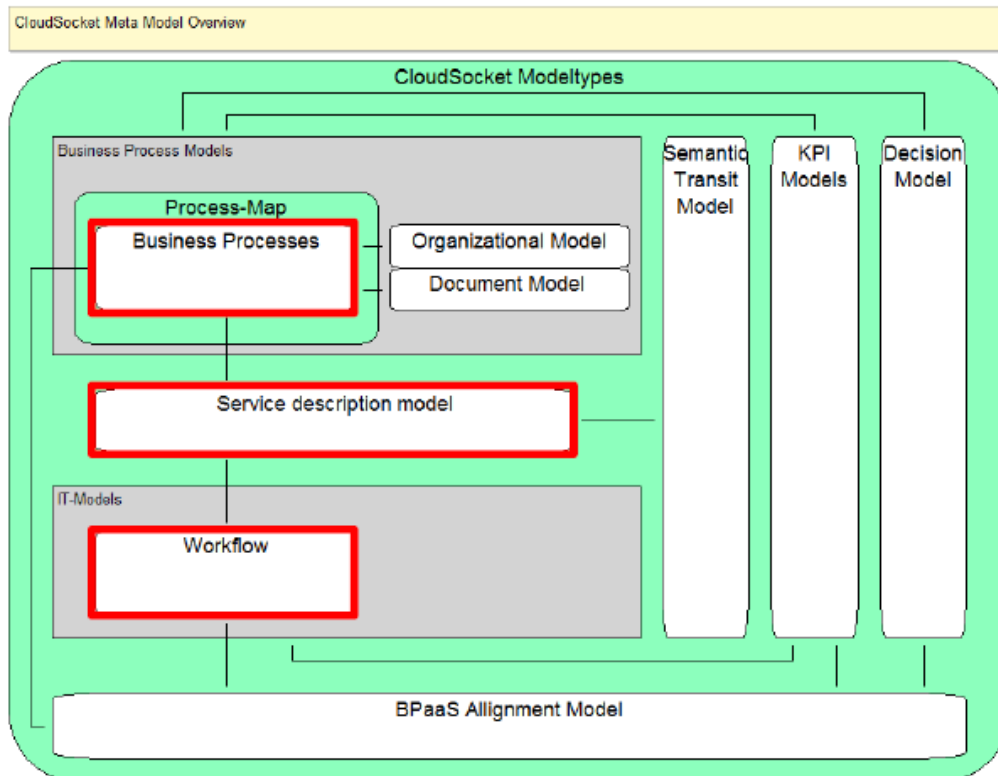


Figure 76. The BPaaS Meta-Models

The three blocks bordered in red colour in Figure 76 are of relevance for this work. Namely, business processes, from the business layer, workflows from the technical layer and the service description model. These were mainly subject to changes during the project whereas the rest model types (e.g. Organisational Model and Document Model) remained unaltered.

For BPaaS DSML, the BPMN 2.0 class diagram was considered and an overview is depicted in Figure 77. The DSML spans to other modelling languages such as the Organisational modelling language and the Document Knowledge modelling language (for documentation see in Appendix B: Business Process as a Service, folder B1), which are integrated with the subset of BPMN 2.0. Like for DSML4PTM, the integration among different modelling languages occurred through bridging concepts, which allowed navigating between concepts of different modelling views.

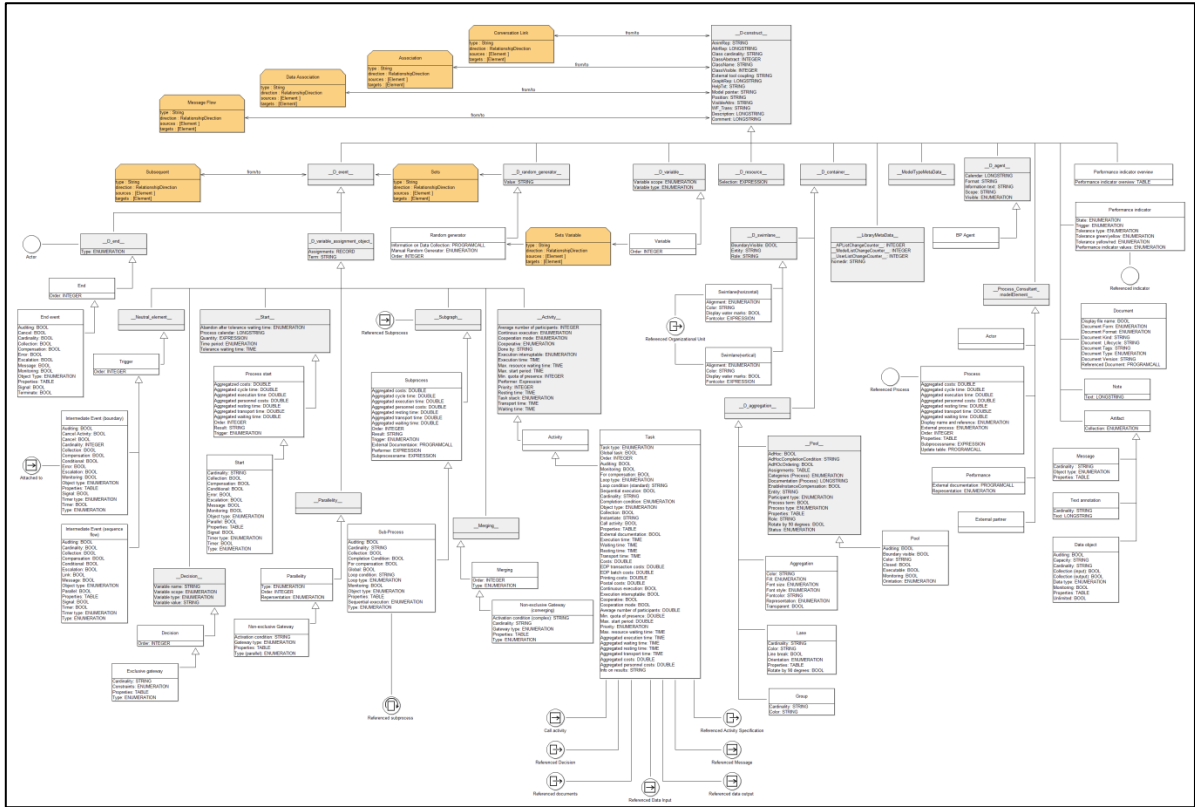


Figure 77. BPaaS Business Process Modelling Language Class Diagram

Business processes and IT workflows can be modelled with BPMN 2.0. Each of them foresees a different modelling view, i.e. the business process modelling view and the workflow modelling view.

To support the matchmaking between the business and IT layer, the modelling standard BPMN 2.0 was extended with a cloud-specific description concept. The new modelling concept was named *Activity Specification* and belongs to the *Service Description* modelling (see red-bordered box between the business process and the workflow one in Figure 76). The concept contained new attributes describing the requirements derived from the business process for cloud services. Nearly 50 attributes were created, e.g. *Name*, *Description*, *General Functionality*, *Payment*, *Costs* (see also Appendix B: Business Process as a Service, folder B1). These were grouped into several dimensions: one (cloud) *Service Requirement* dimension and six cloud service description dimensions. The latter were as follows:

- a) Functional Description,
- b) Input Description,
- c) Output Description,
- d) Non-functional Description,
- e) Business Description,
- f) Regulatory Description.

At this stage, attribute types were mainly in the form of either free text to allow free description of requirements, or enumeration list.

Once the last phase of the engineering lifecycle was achieved, the language could be evaluated by designing the application scenarios and presenting them to the project experts.

From this evaluation, it raised the need to change most of the free text and enumeration attributes (except for numerical values such as downtime per minutes) into program calls. Such program calls aimed promote consistency between the models and the ontology by facilitating the semantic annotation of models. The semantic annotation of models enables the introduction of smart semantic technologies when using conceptual business process or workflow models. The human-interpretable models are enriched with ontology concepts in order to (a) support the alignment between the human- and machine-interpretable representations and (b) to enable business and IT alignment with smart technology. The BPaaS Design Environment architecture in Sub-section 4.2.4 depicts the semantic annotation with an arrow between the ontology and the graphical model. The adaptation requirement is formulated as follows:

Adaptation Requirement 1: *Attributes of the type String should be changed into program calls for semantic annotations.*

4.2.6.3 First Engineering Lifecycle Iteration

To address the *adaptation requirement 1*, the implementation of the meta-model was adapted, i.e. Develop phase. Wherever possible, String attributes were turned into program calls. As afore-mentioned, program calls allowed assigning ontology concepts to attributes, thus semantic annotation of graphical models (see Figure 78). To support the annotation, a Semantic Lifting approach was conceived. In turn, a new deploy of the BPaaS Design Environment was performed.

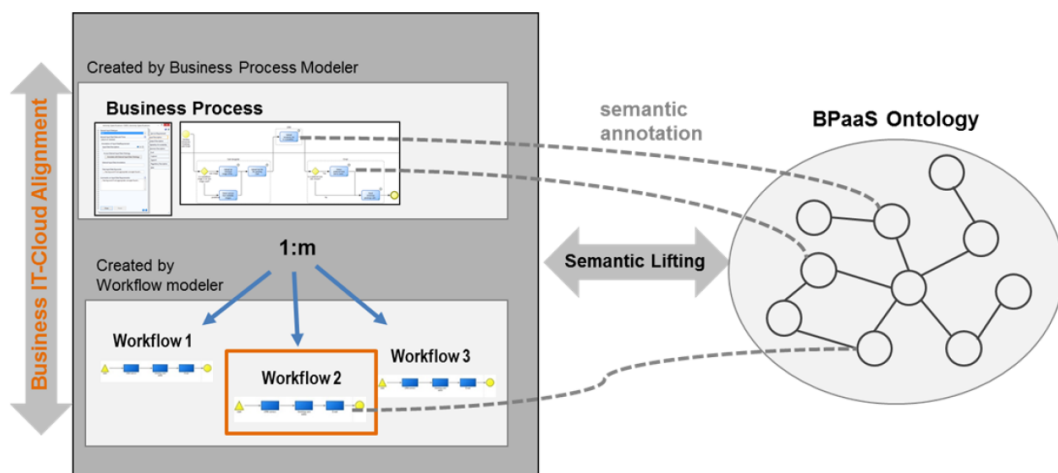


Figure 78. Semantic annotation of models

Following, it is reported an excerpt of the attributes contained in Functional Description dimension. Each attribute is presented with its name, type, (if applicable) range of concrete values, and a brief description.

- (a) The Functional Description dimension (see corresponding implementation in Figure 79) indicates the purpose of the service. It contains the following attributes:
- General Functionality [Enumeration], {not applicable; Apply Rule; Manual; Receive; Transform; Send; Store; Wait; Update; Create; Assign; Others}: defines the expected service behaviour on a very high level. This semantic lifting is a predefined flat list in the form of a drop-down menu.

- Annotation of Functional Requirement: (a) Functional Description [INTERREF], (b) Access External Functional Ontology [PROGRAMMCALL], (c) External Functional Annotation [STRING]. While (a) refers to a particular process task and allows to specify IT- relevant aspects of a task within a business process; (b) enables the selection of ontology concepts representing the functional description. The button invokes an ontology access while the selected concept is stored in the External Functional Annotation (see correspondent textbox populated with the term “invoke” Figure 79).
- Free Functional Keywords [STRING]: In case the afore-mentioned semantic expressions are insufficient or need further elaboration, the author can describe its concerns here.
- Comments on Functional Requirements [STRING]: This feedback enables an evolution based on user feedback.

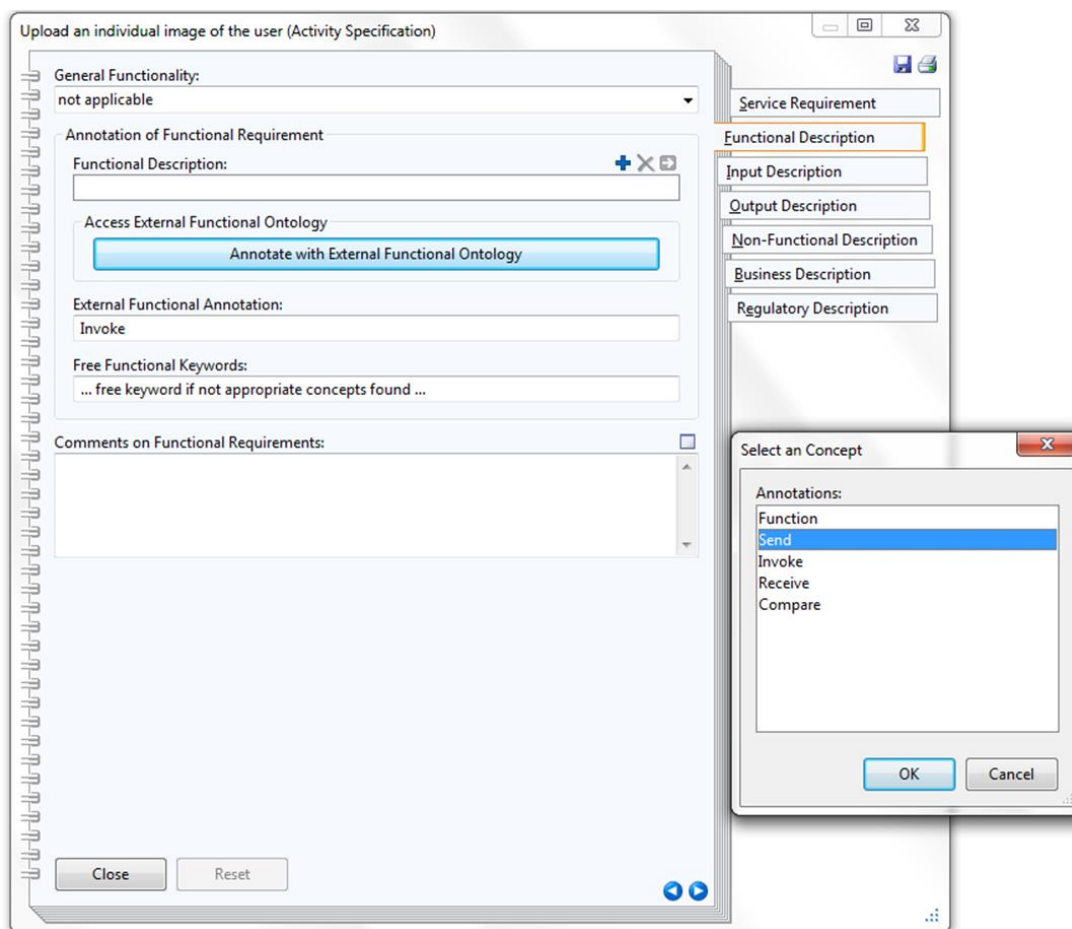


Figure 79. Functional Description dimension implemented in ADOxx

Figure 80 shows the results of the first iteration in the form of a designed model in ADOxx. Namely, the upper part presents the business process modelling view while the lower part the description modelling view (see dashed line in Figure 80). Each process task instantiation leads to one instance of the Activity Specification construct. Arrow 1 points to the Activity Specification construct offered in the palette of the modelling language. Arrow 2 shows the transition from the concrete instantiation of the Activity Specification construct to the attribute dimensions, which in ADOxx are implemented in the form of a notebook (see also Figure 79).

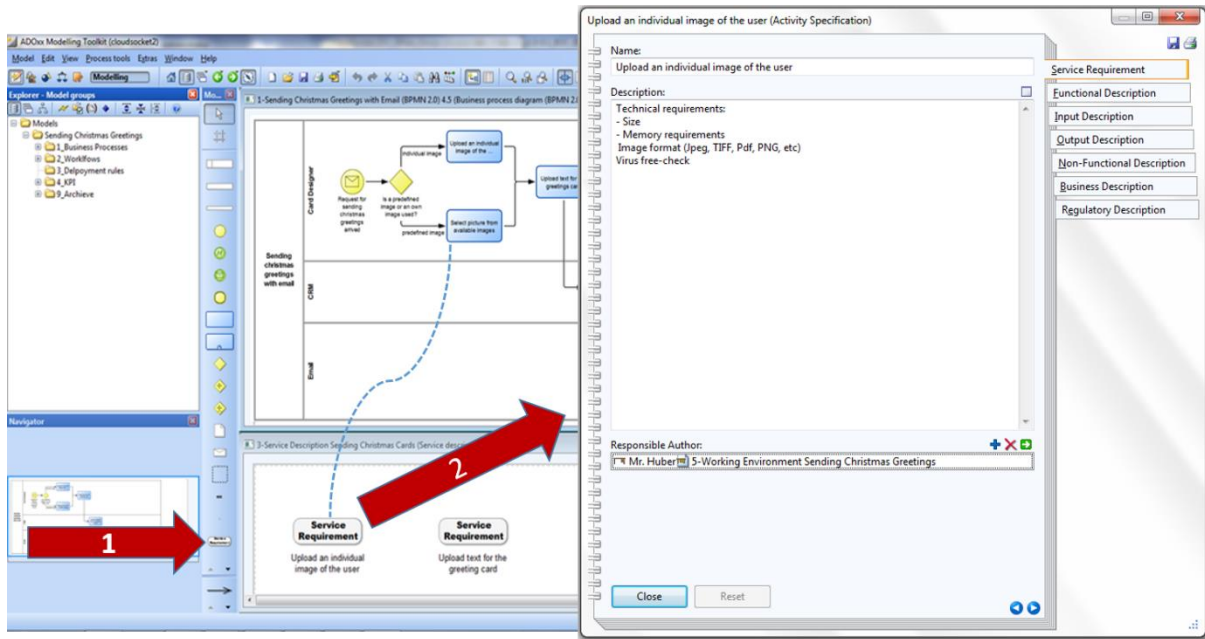


Figure 80. First Iteration: Implementation of DSML BPaaS in ADOxx

Like in the previous version of BPaaS DSML, the evaluation of BPaaS DSML occurred once again by showing models representing the business scenarios to the project members. This activity generated new feedback, which turned into three new main requirements to accommodate.

The first requirement derived from the need to represent (a) business process requirements separate from the (b) IT specifications as well as to provide a modelling view to each representation.

Adaptation Requirement 2: Representations and modelling views for business process requirements should be separate from IT specifications.

The second requirement derived from the need to specify both business process requirements and workflow descriptions not only on a single activity level but rather in different granularity levels, i.e. group of activities and end-to-end business process. This change aimed to provide flexibility in the identification of workflows and cloud services that match business processes requirements. That is, the identification would take place on the basis of requirements related to single activities, group of activities and whole business processes.

Adaptation Requirement 3: Business process requirements and workflow descriptions should be specified over single, group of activities and end-to-end business process.

The third new main requirement was derived by the need to comply with the Cloud Service Level Agreement Standardisation Guidelines (EC Cloud Select Industry Group (C-SIG), 2014) when specifying the non-functional requirements and IT descriptions.

Adaptation Requirement 4: Non-functional business process requirements and IT descriptions should comply with the Cloud Service Level Agreement Standardisation Guidelines (EC Cloud Select Industry Group (C-SIG), 2014).

4.2.6.4 Second Engineering Lifecycle Iteration

To address *adaptation requirement 2*, the “Activity Specification” construct was first adapted. The name changed into “Workflow Description”. Then, a new modelling construct named “Business Process Requirement” was entered. The two modelling constructs specified business process requirements and IT workflow descriptions, respectively. This adaptation implied new attributes in place as well as the change of most of the attributes of the previous version (the complete list of the new attributes can be found in Appendix B: Business Process as a Service, folder B2). For the new list of attributes new standards were consulted (see Service Description modelling views 4.2.6.4.3). Moreover, two separate modelling views were initialized to model on the one hand the business process requirements and on the other hand the workflow descriptions (see Figure 81).

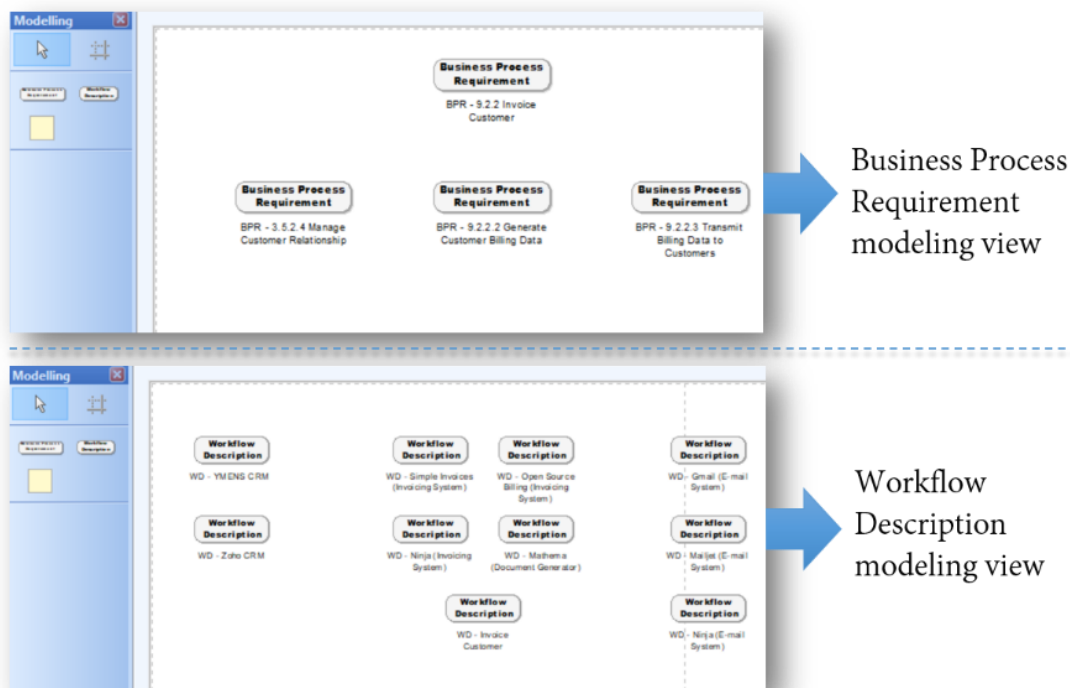


Figure 81. Business Process Requirement modelling view vs. Workflow Description modelling view

In order to navigate through the different views, the following bridging connectors were implemented:

- from the business process modelling view to the business process requirement modelling view,
- from the workflow modelling view to the workflow description modelling view.

The following sub-sections provide a more detailed description of each view and relation between the different modelling views.

4.2.6.4.1 Business Process Modelling View

Differently from the first version, in the second version of BPaaS DSML the BPMN concept “Group” was adapted to add a bridging connector to the “Business Process Requirement” concept.

The connector starts from the process modelling view and leads to the business requirements modelling view. In the latter, the modelling construct “Business Process Requirement” can be instantiated (see upper part of Figure 81). Subsequently, to each “Group” instantiation corresponds one “Business Process Requirement” instantiation. The former is a means for annotating a business process in the business process modelling view. The annotation can take place on a general level, by referring to the whole process, or on a more detailed level specifying requirements for a group of activities, until down to atomic activities (addressing *adaptation requirement 3*).

As an example of the resulting implementation, Figure 82 shows the modelling construct “Group” being instantiated four times for the Send Invoice business process:

- to annotate the whole process,
- to annotate a group of activities: “create new invoice”, “update invoice” and “check invoice completeness”,
- to annotate the single activity “manage customer relationship”,
- to annotate the single activity “send invoice”.

Each group annotation has its own business process requirements, which are conceptualised as attributes in the class “Business Process Requirement”.

In the deployed prototype, these attributes are shown through a notebook interface (see bottom-left part of Figure 82). Section 4.2.6.4.3 further elaborates on the business process requirements.

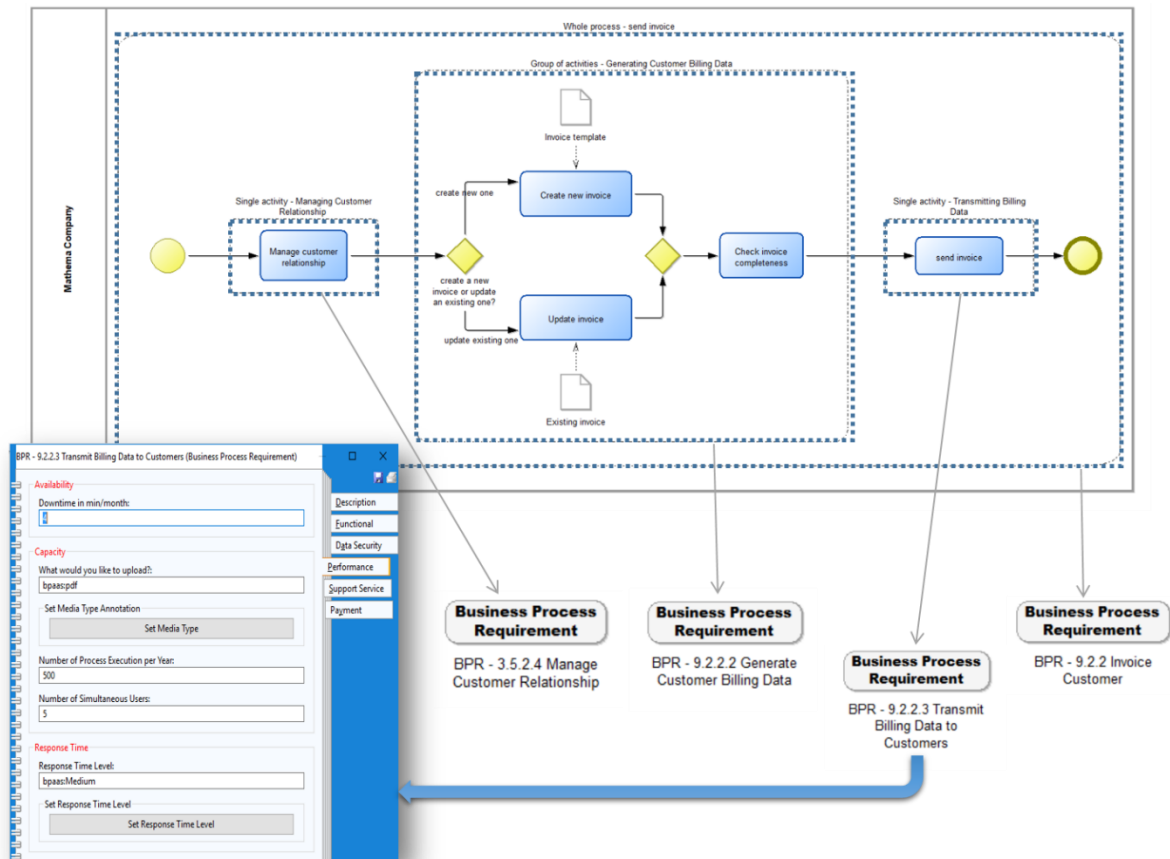


Figure 82. Adaptation of the "Group" concept to specify business process requirements

4.2.6.4.2 Workflow Modelling View

A workflow model reveals technical aspects of the business process and is meant to be designed and understood by technical people. In contrast to business processes, workflows do not have requirements but rather IT descriptions of functional and non-functional properties. As already mentioned, each workflow lane corresponds to an IT system. Hence, the BPMN modelling construct “Lane” was adapted to add a bridging concept leading to the modelling construct “Workflow Description”. Like in the business process modelling view, the latter is instantiated in its dedicated service description modelling view: the workflow description modelling view (see bottom of Figure 81). The three blue dotted arrows in Figure 83 depict the bridging connectors from the workflow modelling view to the workflow description modelling view.

The workflow descriptions are conceptualised as attributes of the class “Workflow Description”. Also, these attributes are shown through a notebook interface (see bottom-right part of Figure 83) in ADOxx. In contrast to the business layer, attributes representing the workflow non-functional specifications are represented in IT terms. This is further elaborated in the “Non-functional specifications” Sub-section of Section 4.2.6.4.3.

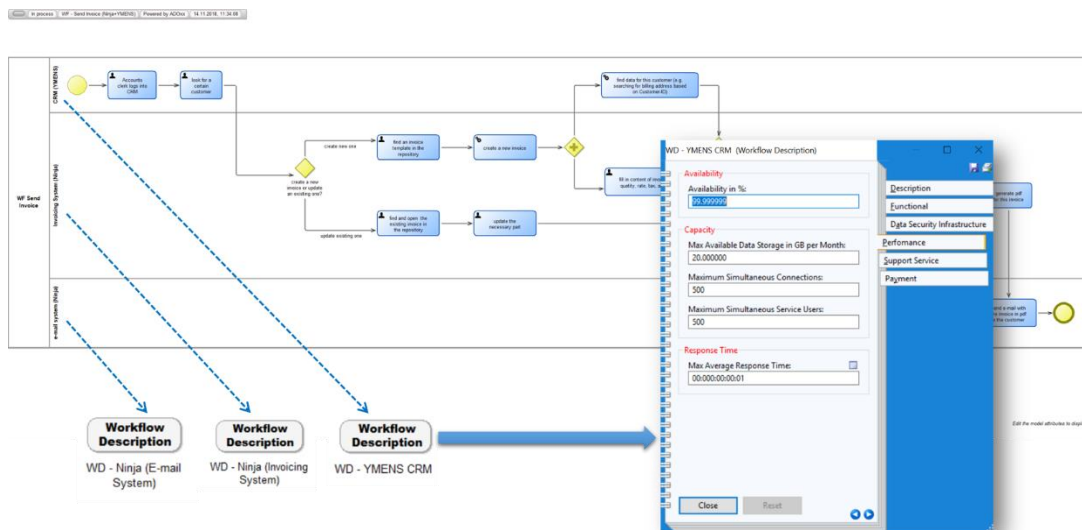


Figure 83. Adaptation of the Lane to specify workflow descriptions

4.2.6.4.3 Service Description modelling views

The Service Description modelling view contains the two above-mentioned modelling constructs “Business Process Requirements” and “Workflow Description”. The former includes functional and non-functional business process requirements whereas the latter functional and non-functional descriptions of workflows. Business and IT properties are distinguished only for the non-functional specifications. The rationale is that business non-functional requirements can be quite different from IT non-functional specifications. For instance, while for a cloud IT expert the meaning of the data security encryption algorithm “Blowfish” would be rather clear, for the entrepreneur who lacks IT background might be not. Rather, the entrepreneur would like to know whether his/her data are securely stored, by mentioning e.g. a percentage or a category level, i.e. low, medium or high. In contrast, specification of a business functional requirement and a functional workflow description can be quite similar as they both express the purpose of either a business process or a workflow, respectively. For example, the purpose of managing invoices can be associated with both a business process and a more technically detailed workflow or cloud service.

Following the functional and non-functional specifications are elaborated.

4.2.6.4.4 Functional Specifications

Functional requirements specify the functionality of a task, a group of tasks or the whole business process. Two ways were proposed to specify the functionality:

- by assigning hierarchies from the APQC Process Classification Framework (APQC, 2014);
- by assigning an action and an object from a predefined taxonomy, which corresponds to the convention of BPMN to name activities by a verb and a noun (Silver 2011). The verb corresponds to the action and the noun to the object, whose combination provide the “what-is-about” knowledge.

In turn, the Functional Description dimension in the BPaaS DSML was implemented with the following attributes:

- APQC
- Action
- Object

To set the value for each of the attributes, an interface is created to the BPaaS Ontology. The interface is invoked by clicking the appropriate button. For example, Figure 84 shows the notebook interface of a Business Process Requirement instance “*BPR – 9.2.2.2 Generate Customer Billing Data*” annotated with the APQC category “*9.2.2.2 Generate Customer Billing Data*”.

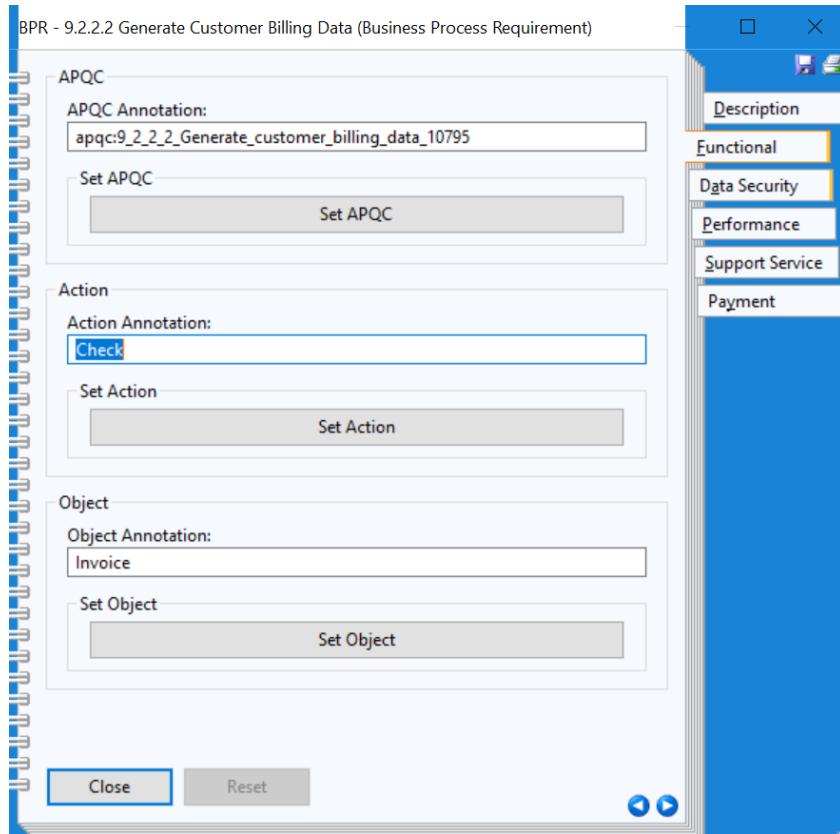


Figure 84. Example of Business Process Functional Requirements annotated with APQC, Object and Action

4.2.6.4.5 APQC Taxonomy

The APQC Process Classification Framework is the most used process framework in the world to communicate and define work processes comprehensively and uniformly within and across organisations. The framework consists of over 1.000 business process categories with an increasing specificity level over a taxonomy of five tiers. As an example, Table 27 shows the first tier with the 13 most generic business process categories.

Table 27. APQC Top Level Categories

1.0	Develop Vision and Strategy
2.0	Develop and Manage Products and Services
3.0	Market and Sell Products and Services
4.0	Deliver Physical Products
5.0	Deliver Services
6.0	Manage Customer Service
7.0	Develop and Manage Human Capital
8.0	Manage Information Technology (IT)
9.0	Manage Financial Resources
10.0	Acquire, Construct, and Manage Assets
11.0	Manage Enterprise Risk, Compliance, Remediation, and Resiliency
12.0	Manage External Relationships
13.0	Develop and Manage Business Capabilities

The highlighted category in Table 27 (i.e. 9.0) is the top generic categories of the more specific APQC category: “9.2.2.2 *Generate Customer Billing Data*”, which was mentioned in the above example.

An ontology reflecting the APQC taxonomy was created (i.e. APQC Ontology) and was used to semantically annotate the APQC attribute. The annotation is supported by the Semantic Lifting prototype, which was developed after going through the first engineering lifecycle iteration. Through a selection box (see Figure 85), the prototype allows the modeller to navigate through the taxonomy tiers from the most generic to the most specific one (and vice versa) until the suitable category is identified.

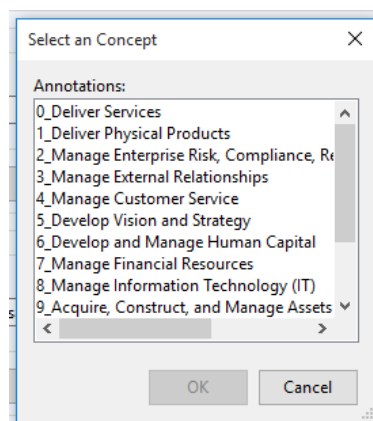


Figure 85. Selection box with the APQC categories of the first tier

Once the suitable category is identified, the modeller can select it and store it as APQC attribute value (as shown above in Figure 84).

Figure 86 shows an excerpt of the APQC Ontology. In the excerpt, the APQC category (“9.2.2.2”) used to annotate the process requirement that is highlighted in Figure 84.

- ▼ ● rdfs:Resource (1800)
 - ▼ ● owl:Thing
 - ▼ ● apqc:AmericanProductivityAndQualityCenter
 - > ● <http://ikm-group.ch/archimeo/apqc#1_0_Develop_Vision_and_Strategy_10002>
 - > ● <http://ikm-group.ch/archimeo/apqc#10_0_Acquire_Construct_and_Manage_Assets_19207>
 - > ● <http://ikm-group.ch/archimeo/apqc#11_0_Manage_Enterprise_Risk_Compliance_Remediation_and_Resiliency_16437>
 - > ● <http://ikm-group.ch/archimeo/apqc#12_0_Manage_External_Relationships_10012>
 - > ● <http://ikm-group.ch/archimeo/apqc#13_0_Develop_and_Manage_Business_Capabilities_10013>
 - > ● <http://ikm-group.ch/archimeo/apqc#2_0_Develop_and_Manage_Products_and_Services_10003>
 - > ● <http://ikm-group.ch/archimeo/apqc#3_0_Market_and_Sell_Products_and_Services_10004>
 - > ● <http://ikm-group.ch/archimeo/apqc#4_0_Deliver_Physical_Products_20022>
 - > ● <http://ikm-group.ch/archimeo/apqc#5_0_Deliver_Services_20025>
 - > ● <http://ikm-group.ch/archimeo/apqc#6_0_Manage_Customer_Service_20085>
 - > ● <http://ikm-group.ch/archimeo/apqc#7_0_Develop_and_Manage_Human_Capital_10007>
 - > ● <http://ikm-group.ch/archimeo/apqc#8_0_Manage_Information_Technology_IT_10008>
 - ▼ ● <http://ikm-group.ch/archimeo/apqc#9_0_Manage_Financial_Resources_17058>
 - > ● <http://ikm-group.ch/archimeo/apqc#9_1_Perform_planning_and_management_accounting_10728>
 - > ● <http://ikm-group.ch/archimeo/apqc#9_10_Manage_international_funds_consolidation_10737>
 - > ● <http://ikm-group.ch/archimeo/apqc#9_11_Perform_global_trade_services_17059>
 - ▼ ● <http://ikm-group.ch/archimeo/apqc#9_2_Perform_revenue_accounting_10729>
 - > ● <http://ikm-group.ch/archimeo/apqc#9_2_1_Process_customer_credit_10742>
 - ▼ ● <http://ikm-group.ch/archimeo/apqc#9_2_2_Invoice_customer_10743>
 - <http://ikm-group.ch/archimeo/apqc#9_2_2_1_Maintain_customer_product_master_files_10794>
 - <http://ikm-group.ch/archimeo/apqc#9_2_2_2_Generate_customer_billing_data_10795>
 - <http://ikm-group.ch/archimeo/apqc#9_2_2_3_Transmit_billing_data_to_customers_10796>
 - <http://ikm-group.ch/archimeo/apqc#9_2_2_4_Post_receivable_entries_10797>
 - <http://ikm-group.ch/archimeo/apqc#9_2_2_5_Resolve_customer_billing_inquiries_10798>
 - > ● <http://ikm-group.ch/archimeo/apqc#9_2_3_Process_accounts_receivable_AR_10744>
 - > ● <http://ikm-group.ch/archimeo/apqc#9_2_4_Manage_and_process_collections_10745>

Figure 86. Excerpt of the APQC Ontology

4.2.6.4.6 Action and Object Taxonomies

As already mentioned, action and object originated from the BPMN convention to name activities by a verb and a noun, respectively (Silver, 2011). Accordingly, an ontology with taxonomies for the both was created and named Functional Business Process Description (i.e. FBPD Ontology). Like for the APQC attribute, the Semantic Lifting prototype supports the semantic annotation for both object and action by retrieving the correspondent ontology. Figure 87 depicts an excerpt from the FBPD Ontology. In the example depicted in Figure 84, the object attribute has the value “Invoice”, which was previously selected from the FBPD Ontology (see highlighted ontology concept in Figure 87, i.e. *fbpdo:Invoice*).

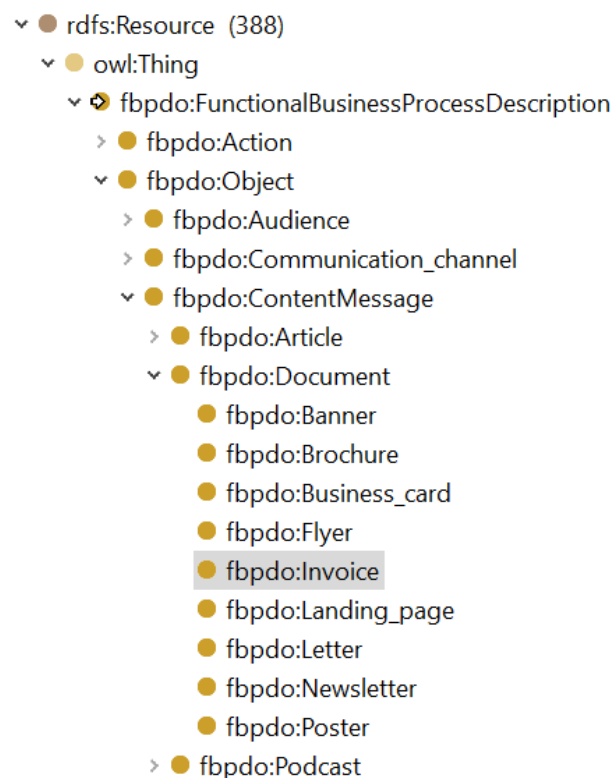


Figure 87. Excerpt of the FBPDO Ontology

Functionality of a workflow and cloud services are specified in the same way with APQC category, action and object.

4.2.6.4.7 Non-Functional Specifications

To address *adaptation requirement 4*, the two kinds of non-functional specifications were proposed: one expressed in business terms (i.e. non-functional business process requirements - NFBPRs), whereas the second one is expressed in IT terms (i.e. non-functional workflow descriptions - NFWDs). Both were derived from the Cloud Service Level Agreement Standardisation Guidelines (EC Cloud Select Industry Group (C-SIG), 2014) and were consolidated in workshops performed iteratively with the project members.

Both categories were grouped into five categories: *Performance*, *Data Security*, *Support Service*, *Payment*, and *Target Market*. For the NFBPRs, each group presents several attributes expressed in business terms, e.g. the performance category includes the media type (e.g. document, video, image and audio) and number of processes executions per time frame (the

complete list of non-functional business process requirements can be found in Appendix B: Business Process as a Service, folder B2).

Similarly, for NFWDs each group presents several attributes, but they are expressed in IT terms. For example, the performance category has *capacity*, which spans the available data storage, simultaneous connections and service users (the complete list of non-functional workflow descriptions can be found in Appendix B: Business Process as a Service, folder B2).

As an example, Figure 88 shows the performance dimension in business terms of the business process requirement element “9.2.2.2 *Generate Customer Billing Data*”, while Figure 89 shows performance dimension in IT terms of the workflow description element “*WD – Ninja (Invoicing System)*”.

The new proposed attributes in the performance dimension (business layer) are as follows:

- *Availability*: Downtime in minutes per month [INTEGER]. It refers to the time when a service is not available.
- *Capacity*: (1) What would you like to upload? [STRING]; (2) Set Media Type Annotation [PROGRAMCALL]; (3) Number of Process Execution per Year [INTEGER]; (4) Number of Simultaneous Users [INTEGER]. (1) refers to the types of media that the user would like to upload in the business process and it allows to store the selected action from the invoked ontology, e.g. pdf documents; (2) enables the invocation of the ontology for the selection of the ontology concept; (3) refers to the number of times a process is executed in a defined period of time; (4) refers to the number of separate cloud service customer users that can use the cloud service at one time;
- *Response Time*: (1) Response Time Level [STRING]; (2) Set Response Time Level [PROGRAMCALL]. Response time refers to the time interval between a call of a cloud service (stimulus) and a response by the cloud service provider. (1) Allows to store the selected response time level from the invoked ontology, i.e. High, Medium, Low; (2) enables the invocation of the ontology for the selection of the ontology concept.

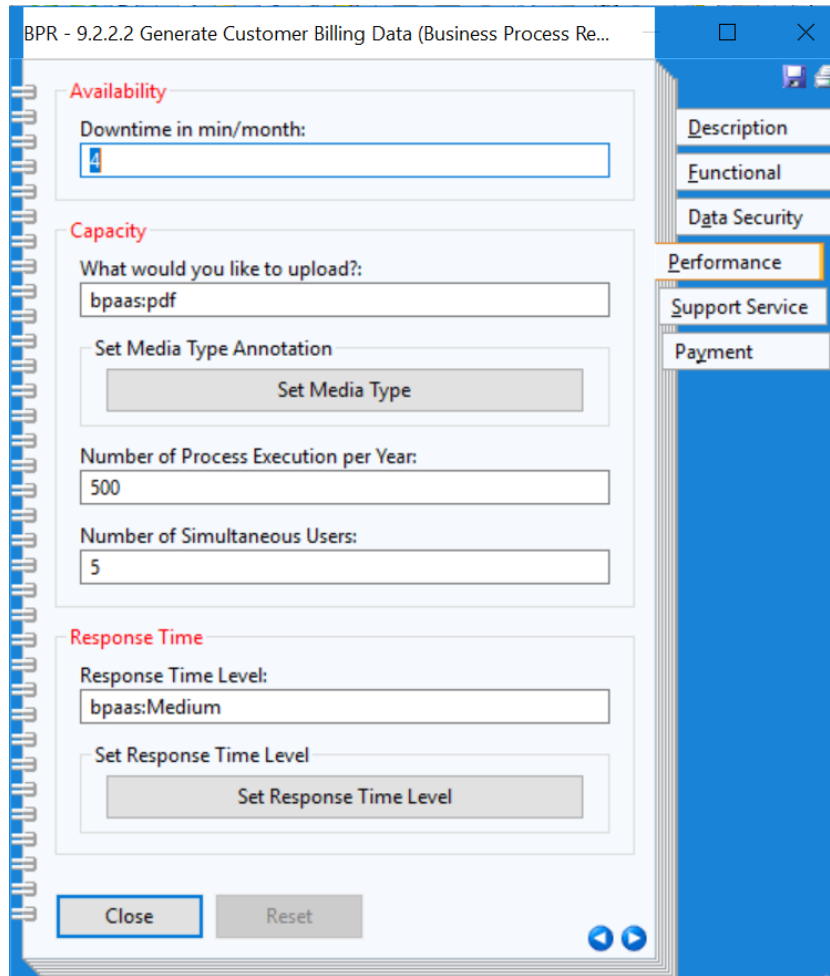


Figure 88. Performance dimension in business terms

The new proposed attributes in the performance dimension (technical layer) are as follows:

- *Availability*: Uptime [DOUBLE]. It describes the time in a defined period (i.e. month) the service is available over the total possible available time and it is expressed in percentage;
- *Capacity*: (1) Max Available Data Storage in GB per Month [Double], (2) Maximum simultaneous connections [INTEGER], (3) Maximum simultaneous users [INTEGER]. Capacity is expressed in terms of both hardware and network. (1) For hardware capacity, the modeller can enter the maximum available data storage in Gigabyte. For network capacity, the modeller can enter (2) the maximum simultaneous connections and (3) the maximum simultaneous users, which is allowed by a cloud service;
- *Response Time*: Max Average Response Time [TIME]. It refers to the time that is required between the requests from the stakeholder to the system and the response of the request.

In this way, workflow specifications are entered by the expert with IT understanding, while business requirements are entered by experts with business understanding.

For example, to specify the data capacity the business modeller can mention the media type he/she wants to upload (e.g. PDF document or a video format) and the number of process executions in a defined time (e.g. up to 500 invoices sent per year). The workflow description

specifies the size of the storage. The comparison between the two languages is made by the smart alignment component, where all models are transformed into an ontology format and given as input to the Matchmaking prototype (mentioned in Section 4.2.4). The latter includes semantic rules and queries to identify which workflow description elements satisfy the business process requirement elements.

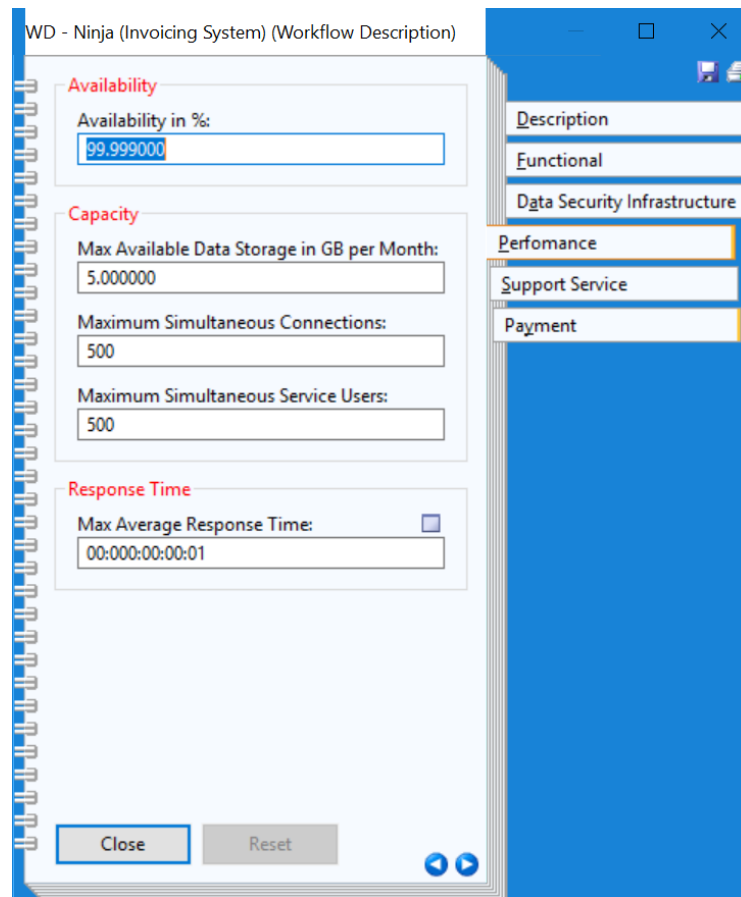


Figure 89. Performance dimension in IT terms

During the evaluation of the prototype, two main drawbacks were identified. One drawback concerned the too much effort for the modeller to specify the business process requirements. Subsequently, the requirement was to specify business process requirements *more intuitively* and *quickly* than annotating process fragments.

Adaptation Requirement 5: *Business process requirements should be specified in an intuitive and quick manner.*

The second identified drawback concerned the non-functional IT specifications. These were limited to the specific use case scenarios. In turn, specifications were further elaborated to reflect as much as possible real-world cloud services.

Adaptation Requirement 6: *Non-functional descriptions should be further specified to reflect real-life cloud services.*

4.2.6.5 Third engineering lifecycle iteration

To address *adaptation requirement 5*, a context-adaptive questionnaire prototype was developed. The new prototype aimed to provide a more intuitive requirement specification from the business perspective. Therefore, process requirements have been expressed in the form of a questionnaire. Answering meaningful and business-related questions made the requirements specification more (business) user-centric. The questions were developed through interviews made with the industrial project partner Mathema⁴⁸, an Italian company operating in the IT field since 1987 and expert in supporting SMEs to identify appropriate Cloud solutions (for documentation see Appendix B: Business Process as a Service, folder B7). In parallel, further findings about non-functional specification of cloud services were included. The findings were derived by the analysis of forty-six recent research papers (from 2009 to 2018) (Giovanoli, 2019). As a result, the following top eight dimensions were derived: *Payment, Security, Performance, Availability, Reliability, Interoperability, Support, Target Market*. The Performance dimension, for example, includes questions mentioned as following:

- Question: What is your preferred monthly downtime in minutes?
 - o Possible answer: 30 minutes
- Question: Should the process be executed on a daily, weekly, monthly or yearly basis?
 - o Possible answer: On a weekly basis
- Question: What is your favourite response time level?
 - o Possible answer: High, Medium or Low
- Question: How many simultaneous users should cloud service support?
 - o Possible answer: at most 10

For each question, four types of answers were distinguished: (1) single-answer selection (i.e. the user selects one answer from a predefined set); (2) multi-answer selection (i.e. the user selects more answers from a predefined set); (3) search-insert; (4) value-insert. Value-insert and search-insert require user input. While the former enables inserting attribute values (e.g. downtime), the latter enables crawling predefined values from the ontology and selecting the suitable one. For instance, answers related to three functional requirement questions (Action, Object, and APQC category) are of the search-insert type. Users can insert keywords for the business process they are looking for, and the ontology returns the concepts matching these keywords. Figure 90 shows this functionality's implementation result, i.e. see a list of APQC categories appearing below the search box.

⁴⁸ <https://www.mathema.com/>

Start New Questionnaire

Question #3
Category: Functional

What does your service do? Please select an appropriate APQC category

search: general

9.2.2.2 Generate customer billing data
3.4.1.3 Generate sales forecast
4.1.3.5 Generate constrained plan
4.3.1.3 Generate detailed schedule
4.3.1.2 Generate line level plan
2.2 Generate and define new product/service ideas
2.2.2 Generate new product/service concepts

9 Matching Services

- Drive
- Dropbox
- Gmail
- InvoiceNinja
- MailJet
- NextCloud
- Ninja_email
- Open_Source_Billing
- Simple_Invoices

Back Next

Figure 90. Search-insert type function applied on the APQC question

Each time a question regarding a business non-functional requirement is answered, a semantic rule is fired either (1) to transform business terms into IT terms, e.g. from downtime to availability in percentage, or (2) to explicate implicit knowledge, thus deriving new facts. This prepares the ground to identify matching cloud services, which is done through a semantic query on a second step. For example, regarding the derivation of new facts (2), let's assume that we have the following:

- A cloud service with the execution constraint of 20 times per day.

Requirements from the filled questionnaire are as follows:

- Should the process be executed on a daily, weekly, monthly or yearly basis?
 - o Answer: At least on a weekly basis.
- How many times should the process be executed?
 - o Answer: At least 10 times

Running a process at least on a weekly basis implies that can also run daily. The semantic rule, therefore, would infer the answer "Daily", which is then used as additional criteria in the semantic query. As a result, the cloud service specification matches the requirement.

4.2.6.5.1 A context-adaptive questionnaire for the quick service identification

In order to provide a quick identification of cloud service (see adaptation requirement 5), the questionnaire was implemented to find the matching cloud service(s) with the least possible number of questions. The questionnaire presents a set of questions that focus first on functional process requirements and then on non-functional requirements, both in business terms. Whereas questions relating to functional requirements are fixed (i.e. same as in the previous engineering iteration: APQC, action, object), questions relating to non-functional requirements are displayed according to a *prioritisation algorithm* (Kritikos, Laurenzi & Hinkelmann, 2018). In a nutshell, the algorithm considers the following:

1. The user preferences in terms of dimensions/categories, e.g. performance, payment, support etc.;

2. The entropy of semantic attributes reflecting cloud service specifications. The higher the entropy value of an attribute is, the higher its service distinguishability degree is, and thus the higher the assigned priority of the related question becomes. This approach leads to the least possible number of questions being posed and answered, thus reducing the requirements-specifications matching time.

The questionnaire is applied on the whole business process first. Next, if no service is found, the user can move down to groups of activities (i.e. process fragments), and lookup services for each fragment. The split and search loop continue until the level of activities is reached and no services have been found for them. The top-down hierarchical service identification is shown in Figure 91; it fosters the identification of services so that they implement as many activities as possible.

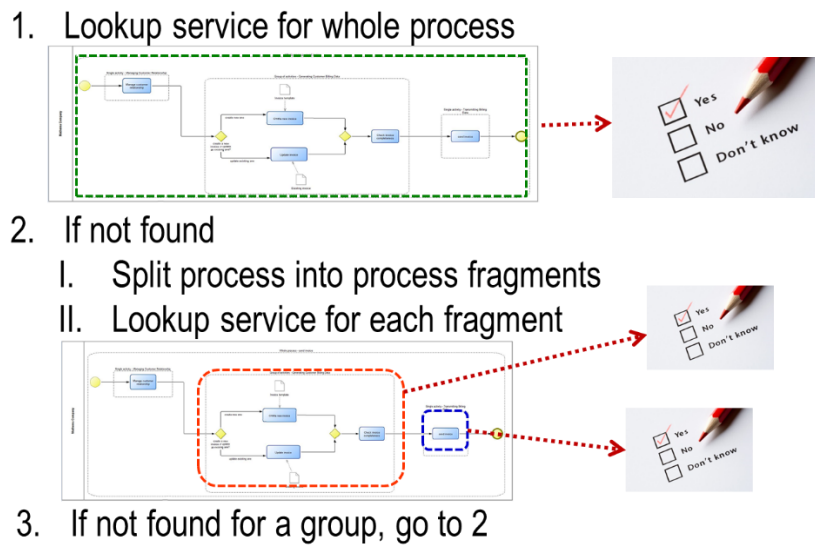
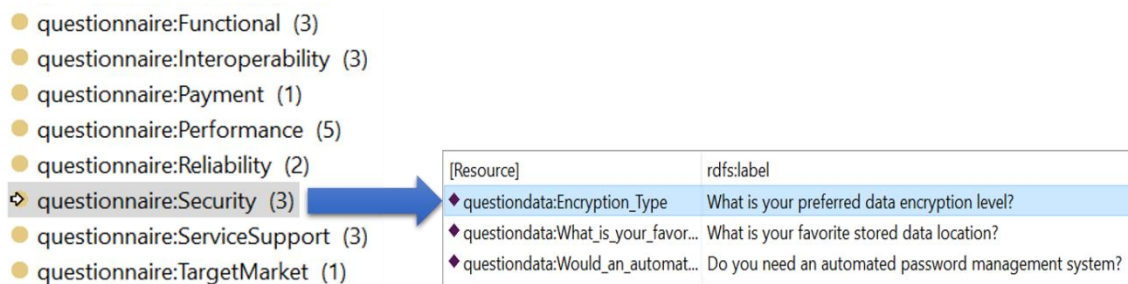


Figure 91. Hierarchical cloud service identification employing the questionnaire

4.2.6.5.2 Ontology-based Meta-Model

The context-adaptive questionnaire adopts the ontology-based meta-modelling approach introduced in (Hinkelmann, Laurenzi et al., 2018). Hence, the semantic alignment between meta-model and ontology is no longer needed and neither the semantic annotation of models. That is, questions and possible answers are displayed in the user interface in the form of graphical representations of concepts that reside in the ontology-based meta-model, i.e. the Questionnaire Ontology. As an example, Figure 92 shows the question that resides in the Questionnaire Ontology in the form of an instance of the concept “Security”, which is one of the non-functional requirement dimensions. The same question is displayed on the user interface of the Questionnaire prototype, see Figure 93.

Figure 92. Excerpt of the Questionnaire Ontology



The matching cloud services are displayed after every answered question on the right-hand side of the user interface of the Questionnaire prototype. For instance, see the two cloud services shown in Figure 93: “Open Source Billing – Invoicing Systems” and “Ninja Invoicing System”.

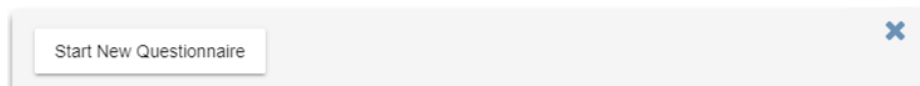


Figure 93. Business process non-functional requirement specification through the questionnaire (Kritikos et al., 2018)



4.2.6.5.3 Non-functional cloud service specifications

To address the *adaptation requirement 6*, non-functional cloud service specifications were re-elaborated. Service descriptions of four marketplaces were screened, i.e. Ymens, IBM, Also and UK digital marketplace. In total, top eight dimensions and related sub-dimensions were identified. Possible values were entered for most of the sub-dimensions. An excerpt of the result is shown in Table 28., where only two top dimensions are displayed (the complete table can be found in Appendix B: Business Process as a Service, folder B8).

Table 28. Non-functional IT Specifications

<i>Top Dimensions</i>	Sub-dimensions	Values
<i>Payment</i>	Payment Plan	Customizable Plan Free of Charge Fixed Subscription Per-terabyte Per-instance Per-user Per-day Per-hour Initial Base Fee Per-Item Utility Pay-as-you-go Monthly Fee None Prepaid Annual Plan Try Free First Other Not specified
	Additional costs	yes no not specified
<i>Security</i>	Encryption type	AES TLS VPN IP Filtering SSL (Secure Sockets Layer) IAAS Ipv6 TLS PSN SOX HIPAA FDA FIPS ISO:27001 SSH HIPAA and HITECH PCI DSS Privacy Shield Other Not specified

4.3 Problems Hindering Agility in DSML Engineering and Main Challenges

As described along with the two cases (Sections 4 and 4.2), seven problems were identified, which hinder DSML engineering. These problems prevented the quick release of new versions of the two DSMLs and are consolidated below (see also Figure 94). Next, the problems are compared with literature findings so to derive the main two challenges to address in this research work.

- *Problem 1:* Language engineers and domain experts have different types of expertise. This might result in misinterpretation of requirements. As a consequence, the quality of the new version of DSML is hampered. This problem manifests particularly in the beginning of the project, as the language engineers have little knowledge about the addressed domain. Further engineering iterations are required until a DSML is good enough to be released, which is time-consuming.
- *Problem 2:* Extracting, documenting, prioritizing, and categorizing requirements is time-consuming and prevents the quick advance of the engineering lifecycle.
- *Problem 3:* The longer the list of requirements and the more dependencies among requirements, the more time-consuming becomes their maintenance (i.e. synchronization and update).
- *Problem 4:* Aligning requirements from the Create phase to the Design phase is time-consuming.
- *Problem 5:* Starting the Develop phase only after the Design phase is finished is not convenient as changes can arise while implementing the language. One reason for changes is due to the inability of meta-modelling tools to accommodate the desired conceptualisation.
- *Problem 6:* The sequential re-iteration of all engineering phases before a new version of a DSML is deployed is time-consuming. That is, a new version of the language cannot be evaluated until the whole lifecycle is ended.
- *Problem 7:* In the Formalize phase, each new requirement originating from the Design Phase (see 7a) or Develop phase (see 7b) leads to adapt the formalized meta-model, which is a tedious, intensive and time-consuming engineering task requiring high expertise.

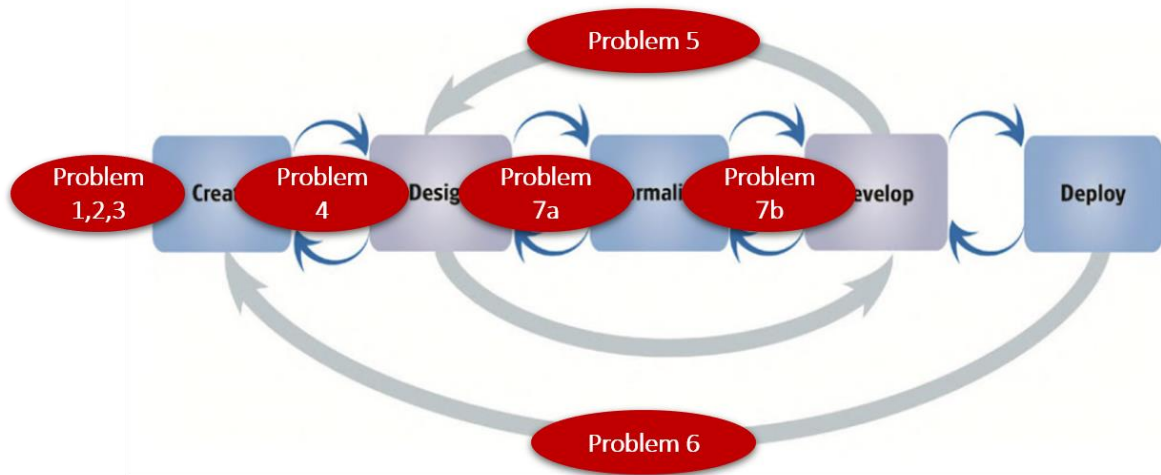


Figure 94. Main problems hindering agility of DSML engineering faced during the development of DSML4PTM and BPaaS DSML

The most significant needs for changes originated from the modelling component when the language could be used as domain experts can provide feedback (see Sub-section 4.1.6.1 for DSML4PTM and Sub-sections from 4.2.6.3 to 4.2.6.5 for the BPaaS DSML). Similar findings are reported by Frank (2013a), who claim that pitfalls related to inappropriate constraints, abstraction issues, or ambiguity of modelling constructs are likely to raise after the language is implemented (see Section 2.9). As shown by the red arrow on the right-hand side of Figure 95, from the Deploy/Validate phase new requirements should be accommodated in the Create phase. Then, the language engineer propagates the new requirements throughout the engineering lifecycle (see right-hand side of Figure 95). After that, the new version of the DSML is deployed so the domain experts can use that. As already mentioned in Sub-section 2.10.4, there are two different components for language engineering and modelling (see also Figure 95).

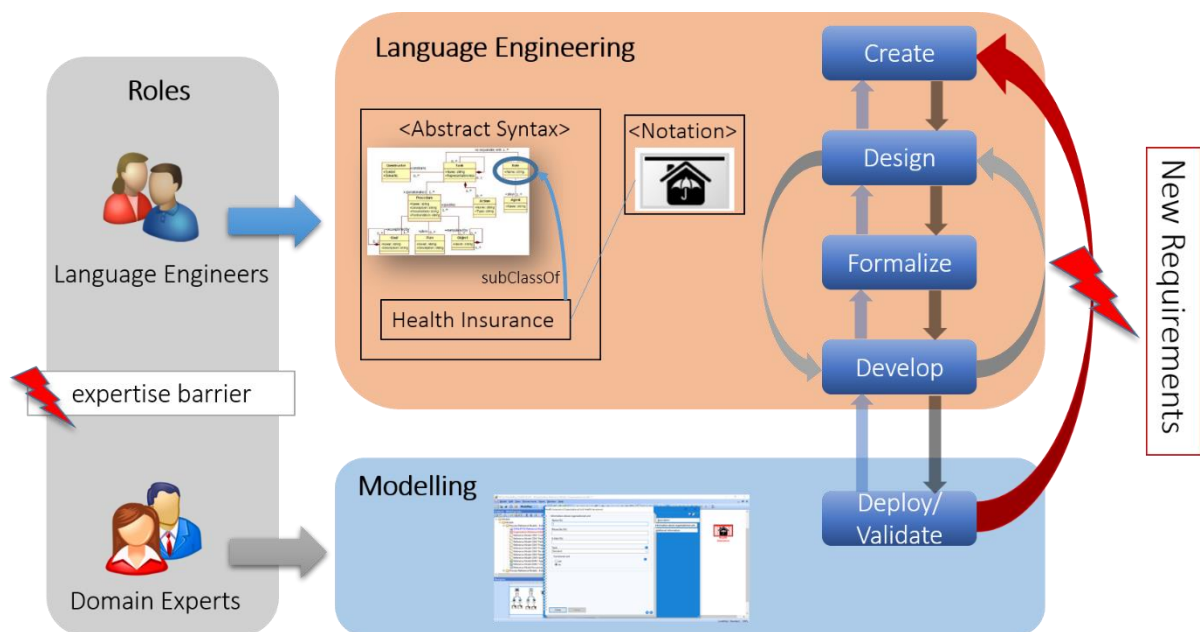


Figure 95. The separation between the language engineering component and the modelling component

Such split leads engineering and modelling activities to be carried out sequentially, thus separately and at different points in time. Domain experts can provide suggestions for adaptations, only after the engineering activity is completed. The suggested adaptations (i.e. new requirements) are not applied on-the-spot as it takes time until a new DSML is released. Therefore, the requirements are gathered and elaborated by the language engineers. However, as pointed in *Problem 1*, such requirements may not reflect the actual needs of the domain experts due to misinterpretations or miscommunications between the language engineers and domain experts. The problem is likely to rise due to the different expertise of the two roles. And also because of little domain knowledge of the language engineers. The little domain knowledge of language engineers is typical in an early stage of DSMLs projects (Frank, 2013a; Barišić et al., 2018). In turn, the quality of the DSML is threatened and new engineering iterations may be required until a version of a DSML is good enough to be released.

A similar problem was identified by Izquierdo et al. (2013), who claim that the inappropriate communication of new requirements from domain experts to language engineers hamper the quality of DSMLs. To overcome this issue, Izquierdo et al. (2013) proposed a collaborative approach between the language engineers and domain experts. They developed an example-driven and collaborative supported tool to engage domain experts in the construction of DSMLs. With a similar end, Barišić et al. (2018) proposed the USE-ME as a methodological approach in which domain experts are involved in the DSML assessment through user interface experimental techniques.

Although these solutions promote the cooperation with domain experts, they are still characterized by a sequential phase, which prevents a DSML from being quickly developed.

Ideally a meta-modelling approach should both facilitate the cooperation between the language engineers and domain experts and avoid the sequential engineering phases while adapting modelling languages (see above *Problem 6*). Thus, the two main challenges to be addressed in this research work are as follows:

Challenge #1: *An agile meta-modelling approach should promote the tight cooperation between domain experts and language engineers by avoiding the separation between the language engineering and modelling components.*

Challenge #2: *An agile meta-modelling approach should avoid sequential DSML engineering phases while adapting modelling languages.*

4.4 Needs for Domain-Specific Adaptations: Experts Interviews

As introduced at the beginning of this chapter, this section describes the answer to the second research question of this work: (RQ2) *What are the needs for domain-specific adaptations of modelling languages?*

In order to answer this research question, four industry experts were interviewed. In the following, a short description of the criteria for the interviewees' selection and expertise is first provided. Then, the interview approach is described followed by a consolidation of interview results. The section ends with a conclusive summary.

4.4.1 Interviewees Selection and Expertise

In order to ensure a high quality of data collection, four experts were selected according to the following criteria:

- Senior position.
- Modelling is a daily business activity to address the client's problems.
- At least two years' experience with enterprise modelling languages in consulting projects.
- Diversity in enterprise modelling expertise, i.e. each expert has a different enterprise modelling focus, e.g. process modelling, enterprise architecture modelling.
- Diversity on the adopted modelling tool, i.e. each expert uses a different modelling tool in project consulting.
- The consultant's company operates world-wide or has world-wide recognized partners.

To differentiate among the four experts, each of them is assigned to a role. In Appendix C: Modelling Expert Interviews, each role is associated with the source document containing the interview transcript.

The first senior consultant is assigned to the role of *enterprise architect* and has more than seven years' experience in IT management projects with a focus on Enterprise Architecture, Business Process Management and Business Analysis. In his daily business, the enterprise architect uses a professional and world-wide adopted modelling tool for modelling enterprise architectures. The modelling is mainly done with the standard ArchiMate. The company of the enterprise architect is an internationally leading manufacturer of software tools for globally recognised management approaches. The modelling tool is owned by the company and allows modelling enterprise architecture.

The second senior consultant is assigned to the role of *business process modeller* and has more than two years' experience in projects with focus on Business Process Management and Enterprise Architecture. In his daily business, the business process modeller uses a professional and world-wide adopted modelling tool for modelling business processes. The modelling is mainly done with the standard BPMN. The company of the business process modeller is an internationally leading manufacturer of software tools for globally recognised management approaches. The modelling tool is owned by the company and allows modelling business processes.

The third senior consultant is head of a consulting area and is assigned to the role of *workflow modeller*. He has more than two years' experience on workflow modelling and execution. In his daily business, the workflow modeller uses a professional and world-wide adopted modelling tool for modelling, integrating and executing business processes, cases and decisions. Respectively, the modelling is done with the standards BPMN, CMMN and DMN.

The company of the workflow modeller offers an open source platform for modelling and automating BPMN business process, DMN decisions and CMMN cases as well as operates world-wide.

The fourth senior consultant is assigned to the role of *enterprise modeller* and has more than eighteen years' experience in project consultancy and training with a focus on Business and IT modelling. Business modelling includes motivation, vocabulary, rule, process, document and organisational modelling. IT modelling includes requirements and executable specifications through Model-Driven Architecture (MDA). In his daily business he uses the company modelling tool for modelling with Object Management Group (OMG) standards such as BPMN, BMM (i.e. Business Motivation Modelling (OMG, 2014a)), OSM (i.e. Organisation Structure Metamodel (OMG, 2006)), SBVR (i.e. for rule modelling (OMG, 2015)) as well as VDML (Value Delivery Modelling Language (OMG, 2018)) of which he is a contributor. The company of the enterprise modeller offers a modelling tool to support developing and running agile enterprises through the model-driven enterprise engineering (MDEE) approach. The company operates mainly locally and is a partner with the Object Management Group (OMG).

4.4.2 Interview Approach

Each interview was conducted and led by myself in bilateral meetings by using semi-structured questionnaire. The questionnaire is divided into two parts: the first part contains five open ended questions while the second part contains questions addressing the suggested artefact of this thesis. This sub-section focuses on the results of the first part of the interview only. Results for the second part of the interviews are described in Sub-section 5.1.6 (the template interview can be found in Appendix C: Modelling Expert Interviews, folder C5).

The questionnaire was sent approximately a week before each meeting to the interviewee. Questions were discussed during the interview and notes were transcribed. Each interview took about two hours. At the end of each interview, the collected notes were further elaborated and transcribed the result in the questionnaire template. The filled questionnaire was then sent back to the interviewee via e-mails to (a) confirm answers, (b) to enter additional information they were specifically asked for, and (c) to edit the transcribed answers. Each questionnaire was then returned, from which PDFs were created (the filled questionnaires in PDF format can be found in Appendix C: Modelling Expert Interviews, folders from C1 to C4).

The interview approach was agreed beforehand and was appreciated by both parties for the following reasons:

- 1) Sending the questionnaire beforehand allowed each interviewee to prepare and thus to start the interview with a common baseline. The common baseline led to efficient and focussed interviews.
- 2) The transcription of notes in the questionnaires made it easy to collect knowledge from experts in minimum time. And interview time was wisely utilised to extract relevant information.
- 3) The three-step approach of letting experts go through the questionnaire before, during and after each interview allowed to increase awareness of the matter, which resulted in insightful answers.

4.4.3 Interviews Results

The questionnaire starts by asking for the problems that clients face when using a standard modelling language in their respective modelling environments:

1. **Question 1:** *Did users face difficulties in learning standard modelling languages and using them in “name of the tool”?*
 - a. *If yes, what are the reasons for that?*
 - b. *How could this issue be overcome?*

Finding 1: According to all four experts’ past experiences, all clients have difficulties in learning and using modelling standards.

Finding 2: Modelling standards are generally too complex for most of the clients to understand and even more challenging to be used. The following three main reasons raised from the interviews:

- The process modeller asserts that standards like BPMN have too many modelling constructs (i.e. elements and relations). In the same line, the solution architect claims that standard languages cover too many aspects that are not needed as project have specific needs.
- The semantics of modelling standards is also problematic for the solution architect. Semantics includes not only the meaning of a modelling construct (i.e. elements and relations) but also constraints, i.e. which elements can or cannot relate to others and by which relation. Both the solution architect and the workflow modeller say that modelling standards have a very broad semantics. The solution architect adds that clients are typically not familiar with such broad semantics.
- The solution architect emphasises the mismatch between the level of abstraction of modelling standards and the clients’ expertise. He claims that ArchiMate is rather technical for business people to understand. In line with it, the workflow modeller states that typically business clients perform worse than IT clients when modelling with standard languages. The workflow modeller associates the reason for it to the logic-driven characteristic of these languages, for which IT clients are more predisposed than business clients. He adds that sometimes case-driven logic like CMMN remains difficult for IT clients as well. In contrast, DMN is a standard evenly understood by both business and IT clients.

Finding 3: To overcome the listed problems, the following three approaches were identified as valid and used approaches:

- *Simplification of the meta-model.* The solution architect, the workflow modeller and the process modeller, state that the initially above-listed problem can be addressed by simplifying the meta-model of the modelling standards, by removing unnecessary modelling elements and relations. For instance, the workflow modeller stresses that his modelling tool does not implement the BPMN parallel start event.
- *Adopting non-standard modelling languages.* According to the solution architect, another possibility to overcome the complexity of modelling standards is to adopt a non-standard language. Language that is built on past projects experience as well as embed some aspects from modelling standards. The expert, in this case, assesses the requirements derived from the client-side (i.e. project and stakeholders) and pick off the shelf non-standard modelling languages and mechanisms matching the given requirements. This approach is, however, less and less adopted as it fosters the creation

of non-standard models, which can only be used (and perhaps re-used) within the same project or company. Hence, model exchange with other companies (e.g. project partners) becomes problematic, as models would be difficult to interpret. Big companies tend to avoid this approach, whereas in SMEs might still go for it.

- *Providing training and documentation to clients.* The solution architect states that training and documentation are provided to clients at the beginning of each project. Documentation includes a predefined vocabulary containing the meaning of relevant terms. For the solution architect such an approach tackles the second and third above-listed problems. In this respect, the enterprise modeller says that in his company, they provide a controlled amount of learning to clients by considering their initial skills. That is clients' knowledge increases in a piecemeal fashion by adding complexity in each subsequent workshop. Similarly, the workflow modeller states that his company offers training that typically consist of two-day workshops where the standard is taught through best and bad practices.
2. **Question 2:** *Could a DSML address one of the problems identified in question 1? Are there more problems that can be addressed?*

Finding 4: All interviewees are positive about a DSML addressing the problems above identified. Additionally, it was identified that the adaptation of modelling standards is rather needed whereas dedicated DSMLs are not considered useful. Following points elaborate this argument.

- The solution architect considers a DSML as the result of a simplification of a modelling standard, which is often the case in projects that aim to model enterprise architectures.
 - For the process modeller a DSML is undoubtedly eligible to address the complexity of modelling standards. He considers not only the language simplification of a modelling standard as a valid approach but also the extension of meta-models with new relation types and new attribute values.
 - The workflow modeller states that it would be useful to have a tool allowing restricting as well as extending the meta-model. He stresses, indeed, the need to specify existing modelling constructs with domain-specific ones. Thus, he gives the credit to domain-specific adaptation of modelling languages. He does not consider useful, however, the modelling elements built from scratch, which is the case of dedicated DSMLs.
 - The enterprise modeller also does not see a dedicated DSML applicable to address the above-stated problems. On the other hand, he emphasises the company practice of creating DSML by extending UML, which is mainly done through profiling techniques.
3. **Question 3:** *Have there been situations, where the modelling languages were not sufficient and where an adaptation could have made sense? Namely, adapting language constructs to fit a more specific domain?*
- a. *If not, why?*
 - b. *Could a functionality for an on-the-fly customisation of modelling constructs be useful in projects with domain-specific target?*

Finding 5: All interviewees stated that there were situations in which the modelling languages were not enough, and the need of adaptation was either done or it would have been needed.

Finding 6: All interviewees consider useful mechanisms that allow adapting a modelling language on-the-fly. For this, most of them use profiling mechanisms implemented in their own modelling tools. This argument is elaborated as follows:

- The solution architect stated that it was often the case that a modelling language had to be adapted in enterprise architecture consulting projects. The adaptation occurred by simplifying the language by making modelling constructs invisible, by customizing the graphical notations, or by specifying modelling constructs, (i.e. creating sub-classes), by specifying or restricting attribute types as well as by assigning pre-defined values. For this, the modelling tool he works with implements the profiling mechanism, which complies with the ArchiMate standard.
 - The process modeller claims that the modelling language is adopted in every business process consulting project. Like in the case of the solution architect, the adaptation refers mainly to the restriction of the meta-model. One example of performed language adaptation for BPMN is such that the process hierarchy and process landscape can be modelled, which are not foreseen by BPMN. Another needed adaptation refers to the integration of the modelling relation “cross-reference” in the BPMN meta-model. Such modelling relation allows linking modelling elements that belong to different models. The process modeller brings two further examples of language adaptation: one about the Swiss government standard “eCH”⁴⁹, in which BPMN was simplified and adapted featuring model guidelines for modelling objects. The other example refers to the “Analysis Meta-Model”, which was adapted to comply with the new European “General Data Protection Regulation” (GDPR) in data protection and privacy. An example of a model with the adapted modelling language can be found in Appendix C: Modelling Expert Interviews, folder C3.
 - The enterprise modeller brings a further example of a modelling standard adaptation. Through the profiling mechanisms implemented in their modelling tool, they extended the Value Flow Modelling (VFM) to model organisational elements such as organisation, organisational units, positions, roles and persons. The extension was required to model who can produce or consume value. Additionally, they implemented some domain-specific extensions in the tool, to model the railways application domain.
 - The workflow modeller never performed modelling language adaptations, although there would have been the need. The modelling tool he works with does not implement profiling mechanisms like the other three tools. He stated that there were situations in which a modelling language needed to be adapted, especially in projects targeting the IT-systems, Pharma and Finance domain. For instance, in his past projects he would have needed to specify modelling elements representing system types like ERP and salesforce.
4. **Question 4:** *For the situations, where the language was not adequate, what kind of modifications on the modelling language would have helped? e.g. creating/deleting/update a modelling construct (i.e. class, attribute and relation on the metamodel level)*

Finding 7: There is a common agreement on adapting the meta-model according to the possibilities offered by profiling mechanisms. Below a list of the recommended changes by each expert is provided:

⁴⁹ <https://www.ech.ch/vechweb/page>

- *Creating and updating elements and relations extensions*, e.g. Compliance Task as a stereotype of BPMN Task (the enterprise modeller, the solution architect)
- *Creating and updating attributes* (the solution architect), e.g. by changing name of a class (the solution architect); adding properties to modelling elements like tags or stereotypes (the enterprise modeller); integrating aspects of different viewpoints to a modelling element, e.g. a BPMN Task that integrates the concepts of responsibility with priority, enforcement and quality (the enterprise modeller); adding user functionality to modelling element, e.g. a specific consistency check (the enterprise modeller).;
- *Customisation of graphical notations*, e.g. by assigning colours or changing shape, providing visibility to attributes that are in the graphical notation (the solution architect, the process modeller); adding explanatory notes to models (the process modeller).
- *Deleting modelling constructs* (i.e. elements and relations) and *attributes* (the workflow modeller) or make them *invisible* (the solution architect).

In contrast to the workflow modeller, the solution architect advises not to delete elements for two reasons: (1) it can create inconsistencies with respect to the already existing elements and relations in both the meta-model and existing models; (2) it might be useful to use modelling construct later in the project.

5. **Question 5:** *Could you provide at least a use case where domain-modelling adaptation would have made sense?*

Finding 8: The major findings concern not only the need for adaptation but also the demand of an approach that allows the quick adaptations to address the need of rapidly addressing the stakeholders' s requests. This argument is elaborated as follows:

- The solution architect provided model containing aspects extended from ArchiMate. These aspects allow modelling Information Security Management Systems (ISMS) scenarios. The model is retrievable in Appendix C: Modelling Expert Interviews, folder C1. Additionally, he added a comment emphasizing the need for supporting the quick adaptation of modelling languages, which promise to deliver value to different IT and business stakeholders.
- As above mentioned, the process modeller provided the extended meta-model to accommodate GDPR aspects. Additionally, the process modeller stated that BPM consultants are continuously under pressure to adapt modelling languages quickly. This pressure is caused by the need to accommodate requests from different stakeholders' categories. If the modelling language or modelling tool are unable to accommodate such requests, it takes quite some time until the development team inserts them. Therefore, according to the process modeller, an approach that allows developing consistent meta-models quickly while cooperating with the addressed stakeholder would cope with the challenge.
- The workflow modeller provided an example of a dynamic batch scenario where an adaptation was required. He claims it was tough to model this scenario as reactions to dynamics changes in worklist had to be modelled. To overcome this issue, a second diagram was created with a custom "multi-instance" modelling construct, where the adaptation took place in terms of semantics only and not in the notation. In particular, the custom class of multi-instance call-activity has an attribute "list", which is observed for modifications. Instances are started or cancelled accordingly to the modifications of the list. This provides the ability to react to changes in the list (the models can be found in Appendix C: Modelling Expert Interviews, folder C4).

- The enterprise modeller mentioned following use cases in his answers: Safety and Risk Analysis (RIAL), Extended Responsibility Modelling (RACI with rules), Extended Requirements Analysis and Evaluation, SEMAT (Software Engineering Method and Theory), Jackson's Problem Frame Approach (PFA) as well as the domain-specific modelling language in the railway's application domain.

4.4.4 Three Complexity Levels of Modelling Language Adaptations

The analysis of all interview findings led to several commonalities. Such commonalities are categorised into three different complexity levels (see Figure 96):

1. At level one there is the simplification of a modelling language. That means, aspects in the abstract syntax that do not match with the elicited requirements are removed. The abstract syntax refers to the knowledge captured by the meta-model, i.e. modelling elements, relations and attributes (including types and values). Moreover, graphical notations can be made invisible. Such needs for adaptation arose from *findings 3, 4, 6 and 7*.
2. At level two we find the change of abstract syntax and notation. Changing attribute values is mainly used to restrict the possible range they can get, which is a way to constrain the modelling language. Changing the graphical notation means that it can be replaced with a new one. Such needs for adaptation arose from *findings 6 and 7*.
3. Level three adds complexity to level two such that the abstract syntax is extended as well as new graphical notations are provided. That is, the meta-model is extended or constrained, which requires a higher level of expertise than the previous two. The meta-model extension includes also the cross-reference relations for connecting concepts from different modelling languages or modelling views. Such needs for adaptation arose from *findings from 4 to 7*.

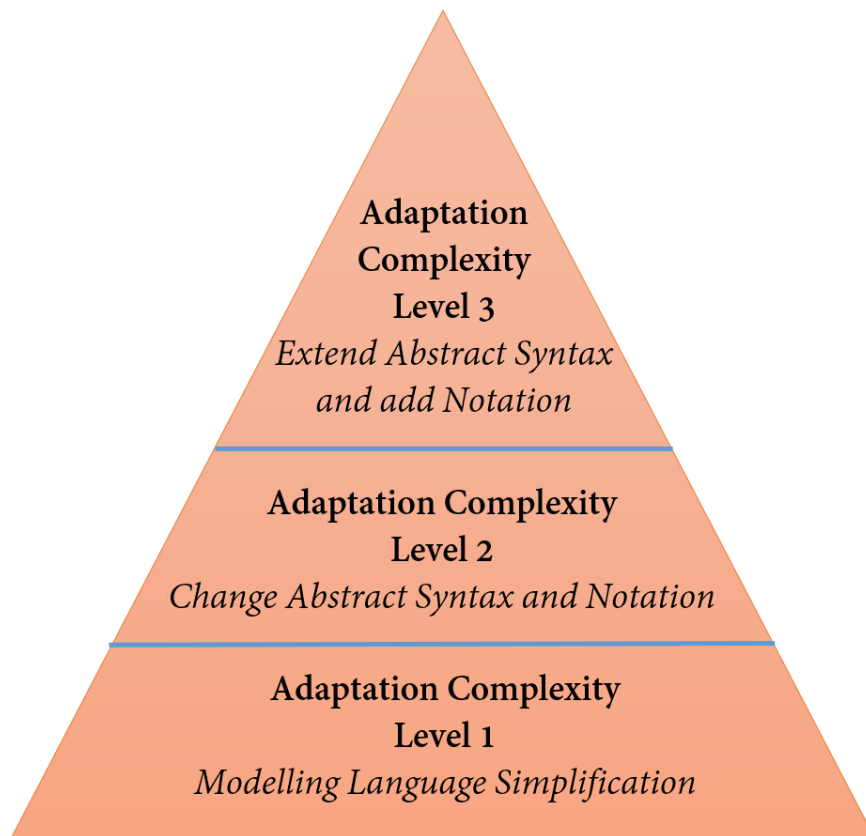


Figure 96. Adaptation Complexity Levels

4.4.5 Requirements for Domain-Specific Adaptations of Abstract Syntax and Notation

This sub-section presents the requirements in terms of domain-specific adaptations to be fulfilled by the new agile meta-modelling approach of this research work. As mentioned in Sub-section 3.6.2, each requirement is conceived with a triangulation approach. The following three sources are considered:

- Complexity levels derived from the interviews' findings.
- Experience from the creation of the two DSMLs (DSML4PTM in Section 4 and BPaaS DSML in Section 4.2).
- Literature review (Chapter 2).

In the following, the list of requirements is described.

Requirement #1: *An agile meta-modelling approach should enable the language engineer to simplify a modelling language.*

The first requirement refers to the language adaptations of *complexity level 1* (see Sub-section 4.4). That is, the new approach should enable the language engineer to (1) delete modelling elements, relations, attributes, and (2) hide modelling elements and modelling relations. The simplification of meta-models by deleting concepts, relations and attributes falls within the common practice of the meta-model customisation technique (see Sub-section 2.8.2). Such practice was adopted in the development of DSML4PTM (Section 4).

Requirement #2: *An agile meta-modelling approach should enable the language engineer to change abstract syntax and notation.*

The second requirement refers to the language adaptations of *complexity level 2* (see Sub-section 4.4). That is, modelling elements, relations, attributes (including types and values) and graphical notations can be changed. Also applying changes in the meta-model is a common practice of the meta-model customisation technique (see Sub-section 2.8.2). This practice was applied for the creation of DSML4PTM (Section 4) and BPaaS DSML (Section 4.2). Karagiannis (2015) stresses the importance of agile meta-modelling approaches to change the meta-model and refers to it with the term *adaptability*. Similarly, Burlton et al. (2017) list perpetual changes as one agile principle in business development (see first principle in Section 2.11).

Requirement #3: *An agile meta-modelling approach should enable the language engineer to extend abstract syntax and add new notation.*

The third requirement refers to the language adaptations of *complexity level 3* (see Sub-section 4.4). That is, new modelling elements, relations, attributes and graphical notations can be added to the existing meta-model. In the two presented cases (Sections 4.1 and 4.2) both meta-models were extended. Karagiannis (2015) stresses the importance for agile meta-modelling approaches to add new aspects to the meta-model and refers to it with the term *extensibility*.

Requirement #4: *An agile meta-modelling approach should enable the language engineer to integrate concepts that belong to different modelling languages or different modelling views.*

This requirement also refers to the language adaptations of *complexity level 3* (see Subsection 4.4). Karagiannis (2015) calls this requirement as *integrability*. The integration can be performed in two ways: (1) by entering a *bridging connector* (or cross-reference) relation between concepts from different modelling languages or different modelling views; or (2) by specializing a class with an existing class from a different modelling language. Both ways were applied to create DSML4PTM (Section 4). In BPaaS DSML, only the bridging connectors were entered to integrate different modelling languages (Section 4.2).

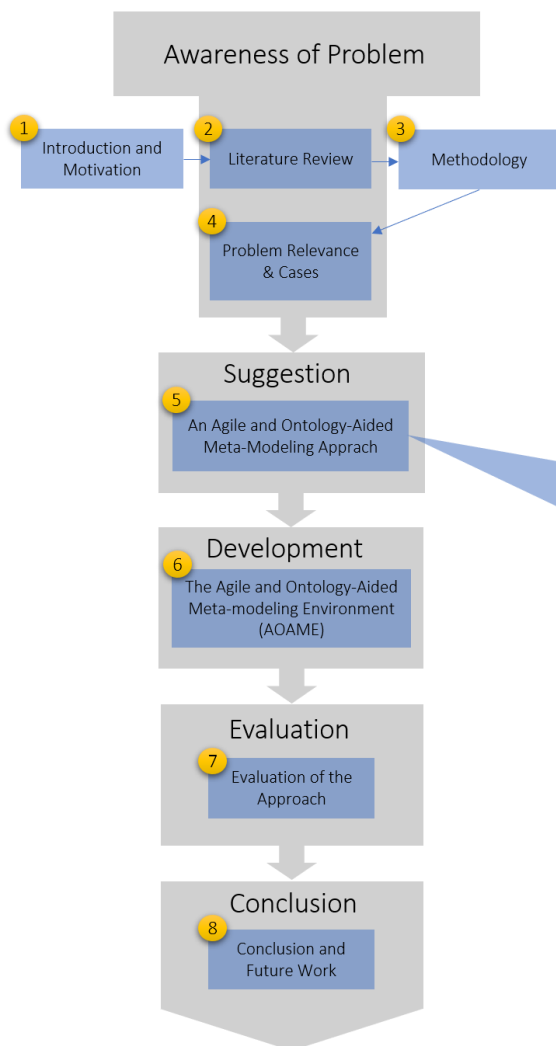
4.5 Concluding Summary

This chapter presented the relevance of the problem addressed in this research work. For this, first two cases were described in which two DSMLs were created through a domain-specific adaptation approach. The two DSMLs were engineered through the AMME Lifecycle. The latter strives to embrace agile principles and is the most advanced in this respect. To increase the generalizability of the problems, the two cases address two different application domains: Patient Transferal Management and Business Process as a Service.

Based on the analysis of the two cases, problems hindering agility of DSML engineering were raised. The problems led to the formulation of two main challenges to address by a new agile meta-modelling approach. Therefore, the first research question was answered.

Then, understanding of the needs for domain-specific adaptations of modelling languages was increased through expert interviews. Interview results were consolidated, and patterns were identified from which three adaptation complexity levels were conceived. Based on the interview findings, literature and experience from the creation of the two DSMLs, a list of four requirements was derived. Like for the challenges, the requirements were proposed to address the design of the new approach of this research work. Therefore, the second research question was also answered.

5. AN AGILE AND ONTOLOGY-AIDED META-MODELLING APPROACH



- 5.1 An Agile Meta-Modelling Approach
 - 5.1.1 Comparison of an Agile Meta-Modelling Approach with Profile Mechanisms for Domain-Specific Adaptations
 - 5.1.2 An Example of the Tight Cooperation in the Integrated Component
 - 5.1.3 Towards the Conceptualisation of On-the-Fly Domain-Specific Adaptations in the Integrated Component
 - 5.1.4 Semantic Mapping of Abstract Syntax to Domain Knowledge
 - 5.1.5 Requirements for the Domain-Specific Adaptations of Semantics
 - 5.1.6 Operators for Domain-Specific Adaptations On-the-Fly
- 5.2 Towards a Machine-Interpretable Semantics for the Agile Meta-Modelling
 - 5.2.1 Semantic Lifting
 - 5.2.1 Problems of Semantic Lifting in the Agile Meta-Modelling
- 5.3 An Ontology-Aided Approach for the Agile Meta-Modelling
 - 5.3.1 The Ontology-based Meta-Modelling
 - 5.3.2 Ontology Architecture for an Agile and Ontology-Aided Meta-Modelling Approach
 - 5.3.3 Designing Semantic Rules for the Propagation of Domain-Specific Adaptations
 - 5.3.4 Syntactic and Semantic Validation of Semantic Rules
- 5.4 Concluding Summary

In Chapter 4, the understanding of the problems hindering the agility of DSML engineering was deepened; and hence, the main two challenges were derived. Understanding the need for domain-specific adaptations was also identified; from which, a list of requirements were elicited. The problems, challenges and requirements are addressed in this chapter with the distinct purpose of designing a new approach for meta-modelling. This answers the third research question of this work:

- (RQ3) *How can agility be fostered when performing domain-specific adaptations of modelling languages?*

The chapter is structured as follows: Section 5.1 introduces the conceptualisation of a new agile meta-modelling approach. The approach includes the introduction of operators for on-the-fly domain-specific adaptation of modelling languages. To support the semantic specification and automation of modelling constructs, Section 5.2 extends the agile meta-model with an ontology-based approach. The approach is then further expanded in Section 5.3 with semantic rules. These rules ensure the seamless propagation of the domain-specific adaptations from the modelling language to the ontology.

5.1 An Agile Meta-Modelling Approach

As described in Section 4.3, the difference in expertise between the language engineers and domain experts could lead to misinterpretations or miscommunications of suggestions for language adaptations (see *problem 1* in Section 4.3). Meta-modelling approaches that separate language engineering from modelling do not overcome this problem (see also left-hand side of Figure 97). In fact the contrary is true; since each component addresses a different expertise, activities for language engineering and evaluation are kept separately and in different points of time. Hence, cooperation between the two roles is not promoted, as the domain expert is not facilitated to intervene until the complete DSML is developed. Promoting the cooperation of the domain experts while engineering a DSML is fundamental to developing a high quality DSML. Therefore, the first challenge was formulated:

Challenge #1: *An agile meta-modelling approach should promote the tight cooperation between domain experts and language engineers by avoiding the separation between the language engineering and modelling components.*

Moreover, it was identified that sequentialising model engineering and modelling prevents the quick development of DSMLs (see *problem 6* in Section 4.3). Hence, the second challenge to be addressed in this research work was derived:

Challenge #2: *An agile meta-modelling approach should avoid sequential DSML engineering phases while adapting modelling languages.*

In order to address the two challenges, an agile meta-modelling approach is proposed. An agile meta-modelling approach is conceived to integrate language engineering and modelling into one component (see right-side of Figure 97). Thus, activities of language engineering, modelling and evaluation can be interleaved and iterated. Such an approach addresses the first challenge as it creates the conditions the language engineer and domain expert to tightly cooperate while engineering a DSML. To address the second challenge, language adaptations are performed *on-the-fly*. That is, language adaptations are applied in a piecemeal manner as demands arise. While language engineers apply adaptations on the DSML, domain experts can visualise the results in real-time, and thus they can provide immediate suggestions. The

sequential approach is therefore avoided as modelling and evaluation activities do not have to wait until a new DMSL is deployed to be performed.

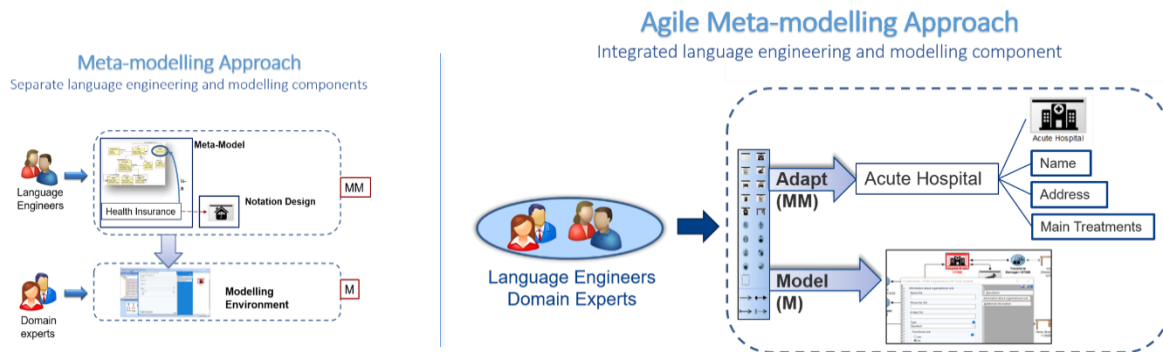


Figure 97. Meta-modelling vs. an agile meta-modelling

Such an approach is also conceived to avoid propagation of changes throughout the whole engineering lifecycle, i.e. *Create, Design, Formalise, Develop and Deploy/Validate phases*. New requirements can be directly implemented into the DSML in a manner that means results are immediately available for testing. The accommodation of requirements through direct implementation also overcome other problems identified in Section 4.3:

- the two problems about the documentation of requirements (see *problem 2* and *3*);
- the problem about the alignment between the Create and Design Phase (see *problem 4*);
- The problem about waiting for the design phase to be finished until the implementation starts (*problem 5*).

The formalisation of the meta-model can be done through the direct implementation of the requirements into an ontology-based meta-model. Such an approach overcomes *problem 7* and is further elaborated Section 5.2.

5.1.1 Comparison of an Agile Meta-Modelling Approach with Profile Mechanisms for Domain-Specific Adaptations

Modelling approaches that make use of *Profile* mechanisms (OMG, 2017) are also implemented in one component, e.g. the modelling tool Visual Paradigm (Visual Paradigm, 2018). Profile mechanisms are promoted by the OMG Meta Object Facility (MOF) (OMG, 2016c) and describe a lightweight extension mechanism to the UML. As described in Section 2.8, such mechanisms are also specified in other modelling standards such as BPMN and ArchiMate. Modelling tools that implement profile mechanisms enable an on-the-fly extension of modelling languages.

Specifically, the extension is allowed by the two extensibility mechanisms *stereotypes* and *tagged values* (OMG, 2017). The two profile mechanisms are as follows:

- *Stereotype* is a profile class, which defines how an existing meta-class (i.e. modelling elements and relations) may be extended. Stereotypes specialise meta-classes, thus creating a new modelling construct. Attributes of the extended classes are inherited from the classes being extended (The Open Group, 2017). The new modelling construct therefore has specific properties that are suitable to a problem domain. A new graphical notation can also be introduced for the new modelling construct.

- *Tagged Values* are used to extend the properties of UML. Such properties are attributes of stereotypes. Tagged values allow the user to specify *types* and *values* to be assigned to attributes. Examples for predefined types are Integer, Boolean, Date and Enumeration.

Such mechanisms foresee two characteristics that are relevant for an agile meta-modelling approach:

- on-the-fly introduction of domain-specific modelling elements and specific properties. As soon as an extension is performed the new modelling elements and attributes are available to be used in a model.
- Inheritance of properties from the extended modelling elements.

These characteristics are limited to the extension of the abstract syntax and notation. Instead, from our list of requirements (see Sub-section 4.4.5), domain-specific adaptations also aim to change and delete aspects of an abstract syntax. Thus, profile mechanisms do not fully satisfy an agile meta-modelling approach.

Instead, the proposed agile meta-modelling aims to fulfil all the requirements derived in Sub-section 4.4.5. Therefore, the approach allows for not only the extension of an abstract syntax and notation, but also for changes and deletions to be made on-the-fly. Moreover, the above-mentioned inheritance principle also manifests when extending an abstract syntax through meta-modelling. Therefore, the same principle applies to our agile meta-modelling.

5.1.2 An Example of the Tight Cooperation in the Integrated Component

In the following, an example is described to clarify the idea of an agile meta-modelling.

In the example we assume the request from a physician (i.e. domain expert) to introduce the concept of an acute hospital as a special class of hospital. The language engineer can accommodate such a request by extending the class “Hospital” with a class “Acute Hospital”, in the integrated component (see Figure 98). The two attributes, name, and address are inherited from the class Hospital. New attributes are then added in the new class, such as a list of main acute treatments, and a graphical notation. As soon as the extension is accomplished, the graphical notation of the modelling construct appears among the modelling constructs of the language, which can be selected to be instantiated as a model element.

The physician can now visualise the new concept and use it. In this way, models can be created to include the new concept. For instance, the new concept can be integrated into the context of an organisational model which includes roles and employees. Such straight implementation of the concept and its immediate use creates the conditions for the physician to provide immediate suggestions for adaptations, should he or she have them. Eventually, these suggestions lead to a more accurate representation of the concept. For instance, new treatment activities might be added, or existing ones deleted, as well as potential desired changes to the graphical notation. Thus, the language engineer and the domain expert tightly cooperate to carry out engineering, modelling and evaluation activities until a satisfactory version of the DSML is achieved.

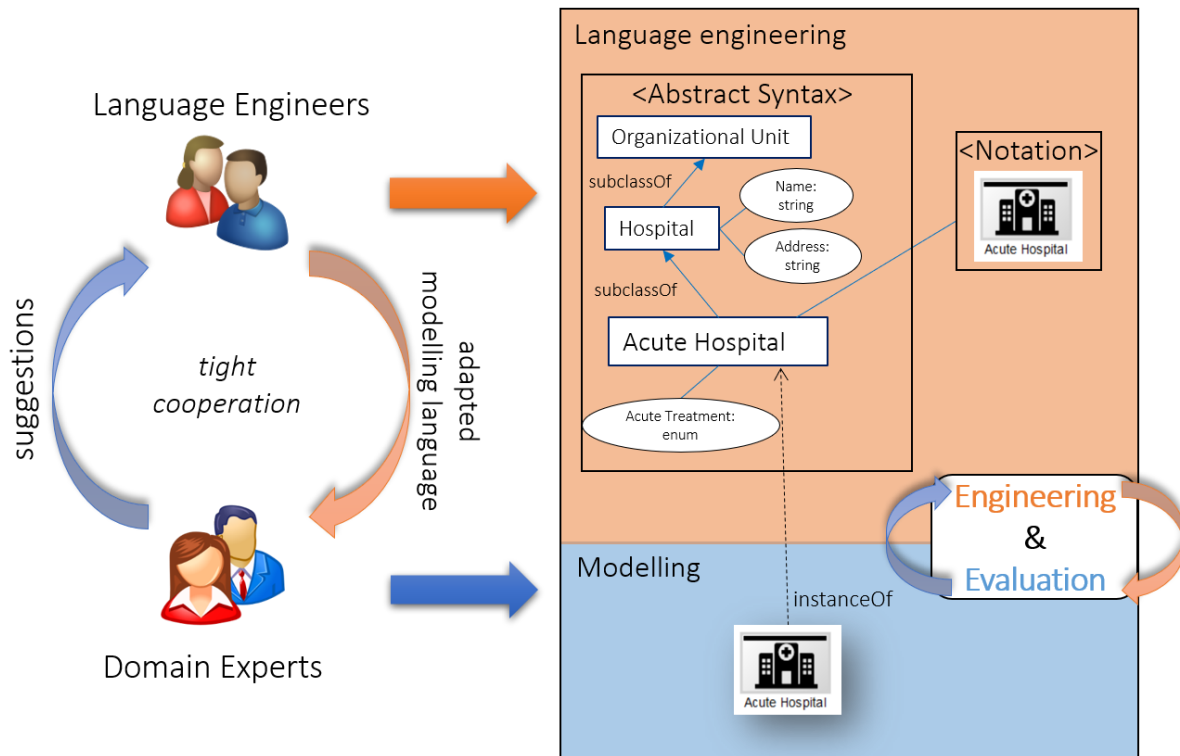


Figure 98. Integrating language engineering and modelling into the same component

5.1.3 Towards the Conceptualisation of On-the-Fly Domain Specific Adaptations in the Integrated Component

on-the-fly domain-specific adaptations are actions that shall be performed in the component that integrates language engineering and modelling. For this, the integrated component foresees a palette (see left-hand side of Figure 99) in which the graphical notations of the DSML are displayed. The palette is conceptualised to provide access to the language engineering component. In the latter, the meta-model can be adapted. Each graphical notation is a gate to the related concept that resides in the meta-model.

For example, the left-hand side of Figure 99 depicts the palette displaying the graphical notations of BPMN. From the BPMN Task one can access the language engineering activities (see orange rectangle in Figure 99, which displays concepts, relations, attributes and notation about the Task). The language engineering activities shall, thereby, allow for domain-specific adaptations of the knowledge residing in the meta-model. For this, the following three specifications should be affected:

- graphical notation,
- abstract syntax,
- semantics.

According to Karagiannis and Kühn (2002), these three specifications define a modelling language (see also Section 2.2).

On the one hand, domain-specific adaptations of a graphical notation and abstract syntax revert to meta-modelling activities such as extending, changing or deleting concepts (see, e.g.,

adaptations performed to create the two DSMLs in Sections 4 and 4.2). On the other hand, the adaptations of the semantics remain unclear as semantics can be found not only in the abstract syntax, but also in a language independent domain, i.e. domain knowledge. In order to suggest an approach that will appropriately adapt semantics, the next sub-section defines semantics, from which adaptation requirements are derived.

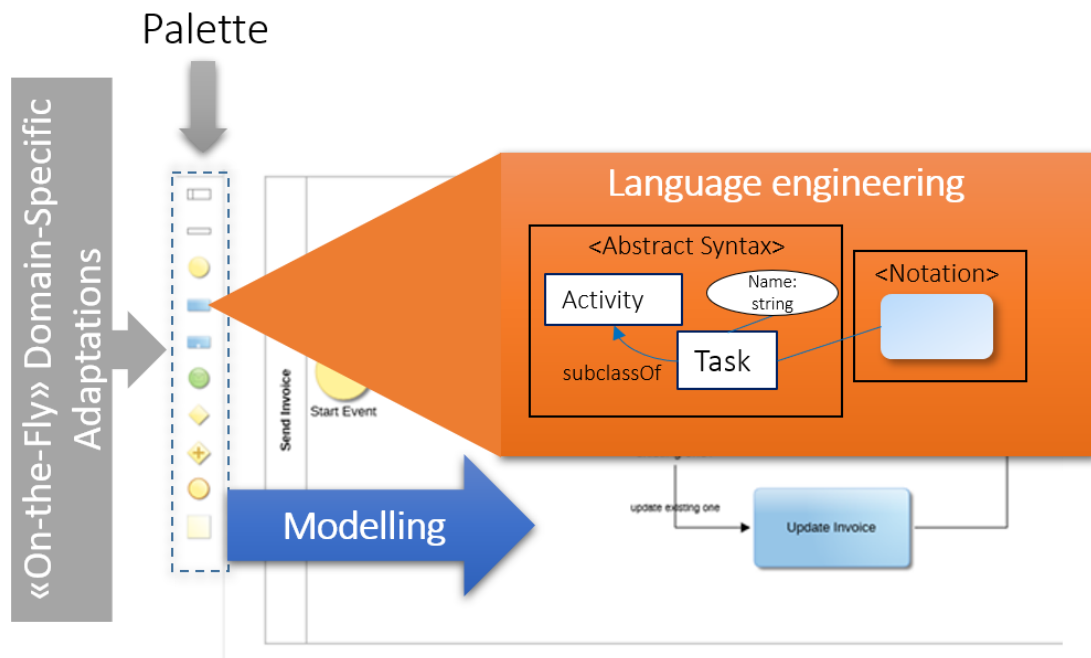


Figure 99. Sketch of an Agile Meta-Modelling

5.1.4 Semantic Mapping of Abstract Syntax to Domain Knowledge

An abstract syntax specifies what kind of knowledge a model can contain. Constraints (or rules) are typically injected in the abstract syntax to govern how the language constructs can be combined to produce valid models. Frank (2013a) claims that the abstract syntax and constraints define the semantics of a modelling language. Such semantic specification is, however, limited to what Atkinson and Kuhne (2003) call the *linguistic view*. Atkinson and Kuhne (2003) stress the importance of not only specifying the linguistic view but also the *domain of discourse view*. The latter describes the *domain knowledge*, which is also known as the *semantic domain* (Harel & Rumpe 2000; Harel & Rumpe, 2004). Harel and Rumpe (2000) state: “The semantic domain can be defined independently of the syntax: in fact, we can use completely different languages for describing the same kinds of systems, so that these languages might all use the same semantic domain”.

Furthermore, they suggest that the semantics of a modelling language can be specified in two parts:

- the *semantic domain*, that provides information regarding the domain of discourse,
- the *semantic mapping*, that maps the abstract syntax into the semantic domain.

The related mathematical formula for the semantic mapping is as follows (Harel & Rumpe 2000):

$$M: L \xrightarrow{\text{maps}} S$$

where the semantic mapping (M) relates concepts from the abstract syntax (L) to the domain semantic (S).

A similar theoretical foundation is also expressed in the framework proposed by Karagiannis and Kühn (2002) (see Section 2.2). As the red arrow in Figure 100 shows, the semantic mapping is connected with the (abstract) syntax. Here, the semantics define the meaning of the abstract syntax by explicating the connection of the semantic mapping.

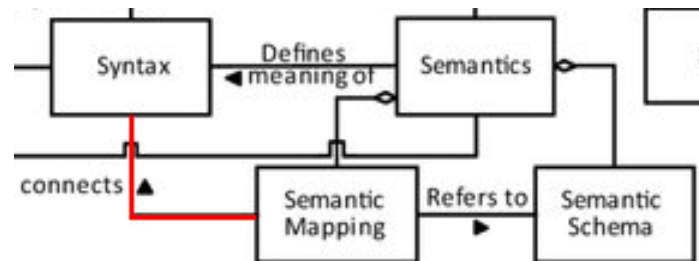


Figure 100. Semantic Specification Components from Modelling Method Components (Karagiannis & Kühn 2002)

Such mapping, therefore, conceptualises the relation between the above-mentioned linguistic view and the domain of discourse view. According to Harel and Rumpe (2000) the semantic mapping should be made explicit. It is not satisfactory to define the semantic mapping by examples as it does not allow for analysis that is sufficient to gain any insight (Harel & Rumpe 2000).

5.1.5 Requirements for the Domain-Specific Adaptations of Semantics

In order to derive requirements for the semantic specification of modelling languages, the semantics of a modelling language should be defined:

1. in the abstract syntax (linguistic view),
2. in the semantic domain (domain of discourse),
3. by mapping concepts from abstract syntax and semantic domain (semantic mapping).

Before deriving the list of requirements, further considerations about the specification of the semantic domain are provided. On the one hand, concepts for the domain semantics might exist off the shelf and come ready to be mapped with concepts from the abstract syntax. For example, this is the case of frameworks or standards that provide categories describing domain-specific and re-usable knowledge, which is independent from any modelling language [(see the International Classification of Functioning, Disability and Health (ICF) (National Center for Biomedical Ontology, 2012) or the American Productivity and Quality Center (APQC))]. Such knowledge can be used to specify the semantic of a modelling construct through semantic mapping. For instance, ICF categories can be used to specify a BPMN data object or APQC categories to specify a BPMN task.

On the other hand, there may be a need to create new concepts for the semantic domain. For example, by adding a new ICF or APCQ category with a higher degree of specificity that is able to better fit the addressed domain. Another case for support is the Business Process as a Service (Section 4.2), namely the extension of the two classes “Action” and “Object” with the classes “Send” and “Invoice”, respectively. Such an extension was done to support the specification of functional business process requirements of cloud services.

In conclusion, the further elaboration of the semantic domain shed the light on the need to not only create the semantic mapping between the abstract syntax and the semantic domain, but also on the possibility to add new concepts for the semantic domain.

On the basis of the additional findings, four requirements are proposed for the domain-specific adaptation of the semantic of a modelling language. The requirements extend the list of requirements for domain-specific adaptations (see Sub-section 4.4.5):

Requirement #5: *An agile meta-modelling approach should enable the language engineer to create new semantic domain concepts.*

Requirement #6: *An agile meta-modelling approach should enable the language engineer to create new semantic mappings between concepts from an abstract syntax (linguistic view) to a semantic domain (domain view).*

Requirement #7: *An agile meta-modelling approach should enable the language engineer to modify the semantic mapping between concepts from an abstract syntax (linguistic view) to a semantic domain (domain view).*

Requirement #8: *An agile meta-modelling approach should enable the language engineer to delete the semantic mapping between concepts from an abstract syntax (linguistic view) to a semantic domain (domain view).*

Note that for the domain concepts there are no delete and modify capabilities. The rationale is that already existing models should not be affected by domain-specific adaptations. If the semantics of concepts are changed, then the models that use these concepts, can become invalid.

5.1.6 Operators for Domain-Specific Adaptations On-the-Fly

In order to perform on-the-fly domain-specific adaptations on one or more modelling languages, a list of *10 operators* has been identified. The operators allow applying changes and extensions over the specifications of a modelling language, i.e. abstract syntax, notation and semantics. The four requirements for the domain-specific adaptations of abstract syntax and notation are described in Sub-section 4.4.5, whilst for semantics an additional four requirements are described in Sub-section 5.1.5. The complete list of requirements is the starting point for the generation of the operators. Figure 101 shows an overview of the steps made to conceive the 10 operators. Namely:

1. Mapping the eight requirements to the CRUD (Create, Read, Update, Delete) paradigm,
2. Deriving the first draft of operators,
3. Refining the list of operators through expert interviews,
4. Publishing the operators in (Laurenzi et al., 2018),
5. Further refining the list of operators according to the reviewers' comments during publication.

The deriving process of the operators is further elaborated upon in Sub-section 5.1.6.1. Then, the operators are introduced (Sub-section 5.1.6.2) and finally a critical reflection surrounding the use of the operators is provided in Sub-section 5.1.6.3.

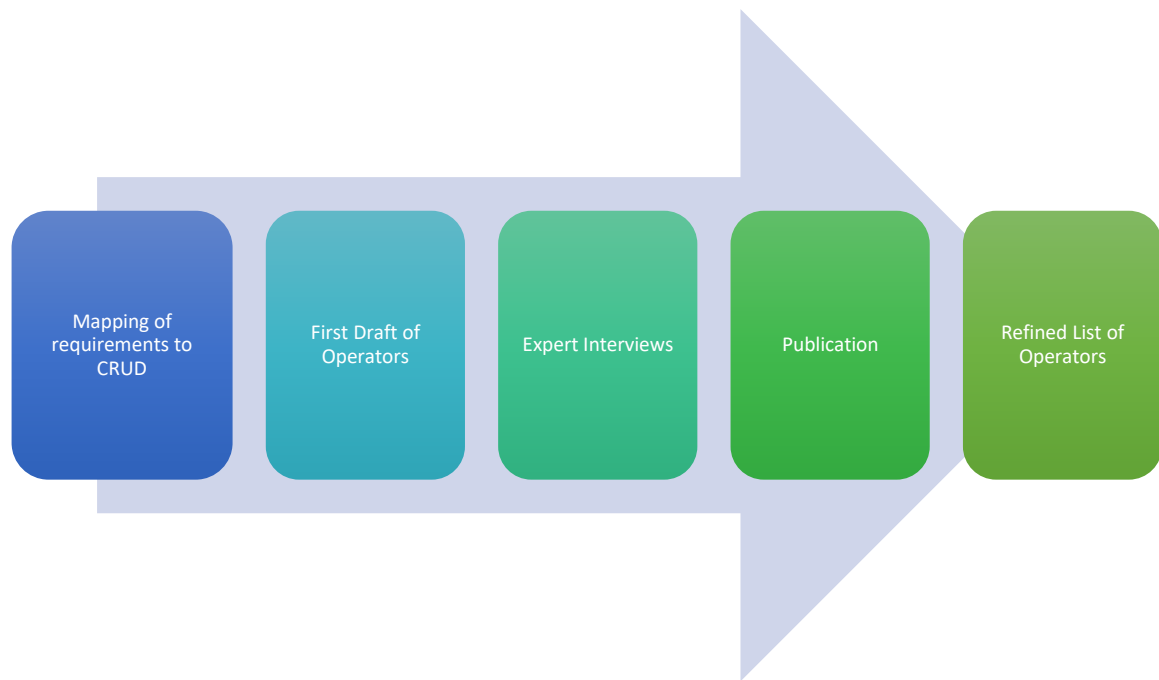


Figure 101. Derivation of Operators

5.1.6.1 Deriving Process for Operators

As already mentioned, the eight requirements were mapped to the CRUD paradigm (Martin & James, 1983). The latter foresees four basic functions, which are implemented in relational database applications: *Create*, *Read*, *Update*, and *Delete*. Each function can map to a SQL (Structured Query Language) statement:

- Create maps to INSERT,
- Read maps to SELECT,
- Update maps to UPDATE,
- Delete maps to DELETE.

Table 29 shows the mappings between the requirements and the basic functions of the CRUD paradigm. Namely, *requirements 3 to 6* can be fulfilled by the function *Create*, *requirements 2 and 7* by *Update*, and *requirements 1 and 8* by *Delete*. There is no requirement mapping to the function *Read*, which is obvious, as the function does not fit the purpose of changing or extending the modelling language specifications.

Table 29. Mapping requirements to basic functions CRUD

Requirement Number	Description	CRUD function
<i>Requirement #1</i>	An agile meta-modelling approach should enable the language engineer to simplify a modelling language.	Delete
<i>Requirement #2</i>	An agile meta-modelling approach should enable the language engineer to change abstract syntax and notation.	Update
<i>Requirement #3</i>	An agile meta-modelling approach should enable the language engineer to extend abstract syntax and add new notation.	Create
<i>Requirement #4</i>	An agile meta-modelling approach should enable the language engineer to integrate concepts that belong to different modelling languages or different modelling views.	Create
<i>Requirement #5</i>	An agile meta-modelling approach should enable the language engineer to create new semantic domain concepts.	Create
<i>Requirement #6</i>	An agile meta-modelling approach should enable the language engineer to create new semantic mapping between concepts from the abstract syntax (linguistic view) and concepts from the semantic domain (domain view).	Create
<i>Requirement #7</i>	An agile meta-modelling approach should enable the language engineer to modify the semantic mapping between concepts from the abstract syntax (linguistic view) and concepts from the semantic domain (domain view).	Update
<i>Requirement #8</i>	An agile meta-modelling approach should enable the language engineer to delete the semantic mapping between concepts from the abstract syntax (linguistic view) and concepts from the semantic domain (domain view).	Delete

Each of the three functions Create, Update and Delete, performs one specific domain-specific adaptation, e.g. create a new modelling element as a sub-class of an existing modelling element, create a new sub-class relation between two existing modelling elements, update the name of a class etc. In this sense, domain-specific adaptations affect one of the following basic generic modelling concepts:

- Class,
- Relationship,
- Attribute,
- Attribute types and values.

Next, the first draft of operators was determined, presented to- and discussed with-modelling experts in the second part of the interview introduced in Section 4.4 (the second part of the template interview can be found in Appendix C: Modelling Expert Interviews, folder C5). The template contains screenshots for each operator. The screenshots contain an early design of an agile meta-modelling approach that helped to visualise the purpose of the operators.

Firstly, experts were asked for critical feedback regarding the operators.

Finding 1: There is a general agreement on the fact that operators make sense. However, each expert adds a personal comment as follows:

- The solution architect discourages the use of operators that lead to the deletion of modelling elements, relations and attributes. The delete operators should only be used on the extended part of the modelling language.
- The process modeller claims that operators that lead to the deletion aspects of the meta-model are critical, as they can create inconsistencies on existing models as well as risk the damaging of the meta-model.
- The enterprise modeller also stresses that removing concepts can have severe consequences to the meta-model. Thus, he suggests that delete operators are only available to a meta-model expert (i.e. language engineer).
- The workflow modeller claims that all the models he designs in his modelling tool contain classes. Classes are then instantiated by the workflow management system. Thus, he never faced the need to distinguish between instances and classes in a model (i.e. operator 11).

Operator 11 was described as follows:

- Operator 11 allows specifying whether a modelling element is a type or instance. Hence, the modelling environment can model instances as well as classes. Since a class can be an instance of another class, the modelling approach enables distinctions to be made between different levels of abstractions that are not restricted to the class-instance dichotomy of a description logics representation. For instance, if we consider the data object in BPMN, depending on the purpose of the process model, there might be the need to represent it as a class or as an instance containing concrete values.

Operator 11 was the only operator that was subject to criticism when publishing the list of operators (Laurenzi et al., 2018). In fact, this operator does not deal with domain-specific adaptations of a modelling language. Rather, it reflects the abstraction issue of a model. Therefore, it was removed as it is out of the scope of this research work.

Critical consequences about operators removing aspects of an abstract syntax are further elaborated in Sub-section 5.1.6.3.

Secondly, the experts were asked to suggest any additional adaption operators and actions.

Finding 2: No further operators were suggested. Three experts suggested some possible functionalities. In detail:

- The solution architect suggests a functionality that hides modelling constructs.
- The process modeller suggests functionalities that insert pictures and notes on a model, as well as those that can easily change the graphical notations of modelling constructs.
- The enterprise modeller suggests the same functionalities as profile mechanisms (see also Sub-section 5.1.1), i.e. changing properties, graphical notations, renaming properties and relationships. Such aspects are already foreseen by the meta-modelling approach. Additionally, he suggests functionalities to be used while modelling such as adding/updating/removing context menu, toolbars, reports).

Among these suggestions, the functionalities aiming to improve the modelling experience remain out of the scope of this research, as the stated aim is not to improve user experience, but instead to develop an approach for domain-specific adaptations. The other suggestions were already provided during the first part of the interviews, which are describe in Sub-section 4.4.3.

5.1.6.2 Operators for on-the-fly Domain-Specific Adaptations

This sub-section describes the final list of operators that consider feedback from the experts. Each operator is associated with a corresponding requirement for the on-the-fly domain specific adaptations (see Table 29 in Sub-section 5.1.6.1). Figure 102 graphically depicts such associations.

Operator 1: Create sub-class. This operator aims to fulfil three requirements. It allows modelling elements and modelling relations to be extended (*requirement 3*). It allows the integration of modelling elements (classes) from different modelling languages (*requirement 4*). It allows extension of concepts in the semantic domain (*requirement 5*).

Operator 2: Update class. This operator allows modifying an existing modelling construct (*requirement 2*), which includes attributes such as name, comment, and graphical notations.

Operator 3: Delete sub-class. This operator allows the removal of unneeded modelling constructs (i.e. modelling elements and modelling relations) from the abstract syntax (*requirement 1*).

Operator 4: Create relation. This operator allows the creation of new relations (i.e. *bridging concept*) between modelling elements in the abstract syntax (*requirement 4*) as well as new relations from modelling elements to semantic domain concepts, i.e. *semantic mapping* (*requirement 6*). Bridging concepts refers to the relations that connect modelling elements between different modelling languages or modelling views.

Operator 5: Update relation. This operator allows the modification of the existing relations between modelling elements as well as relations from modelling constructs to semantic domain concept(s) (*requirement 7*).

Operator 6: Delete relation. This operator allows the deletion of existing relations between modelling elements as well as relations from modelling constructs to semantic domain concept(s) (*requirement 8*).

Operator 7: Create attribute. This operator allows for adding new attributes to modelling elements (*requirement 3*).

Operator 8: Assign attribute type or value. This operator allows for assigning types (e.g. String, Integer, Boolean) or concrete values to attributes of modelling constructs. A concrete value is the reference to a graphical notation (*requirement 3*).

Operator 9: Delete attribute. This operator allows for deleting existing attributes from a modelling element (*requirement 1*).

Operator 10: Update attribute. This operator allows for modifying existing attributes associated to modelling constructs (*requirement 2*), i.e. the name and the value type of the attribute.

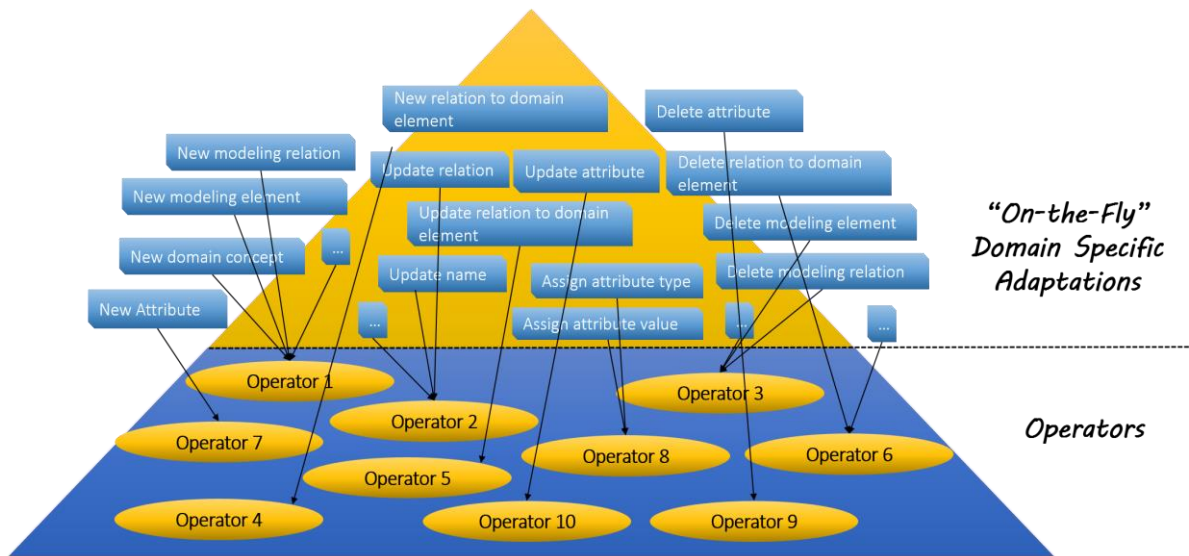


Figure 102 The 10 operators for on-the-fly Domain-Specific Adaptations

5.1.6.3 Critical Reflection on the Use of Operators

Three issues were identified from the use of the operators for domain-specific adaptations:

- (1) One issue regards the operators that implement the function delete, i.e. *operators 3, 6 and 9*. For instance, deleting a root concept in the class diagram (i.e. abstract syntax) implies that sub-concepts will also be deleted, which may be undesired. This issue is tackled by prohibiting the deletion of elements that have sub-elements. Thus, only elements can be deleted. Such logic has been implemented in the Delete functionality and is described in Sub-section 0.
- (2) The second issue regards the inconsistencies that may arise by the operators implementing the functions delete and update with respect to the models already created, i.e., *operators 2, 3, 5, 6, 9*. That is, if a construct is removed or changed from a modelling language, any model that uses that removed construct becomes inconsistent to the modelling language (Rose et al., 2009). Four solutions were identified to tackle this issue:
 - a. Stakeholders shall agree upon a DSML version before creating valid models. Modellers and domain experts have no rights to change the DSML. If changes are needed, the language engineer intervenes on the modelling language and provides suggestions on how to cope with the existing models. *Drawback:* high workload on the language engineer and long wait times while the first valid models are created.
 - b. Propagating alerts throughout existing models that are affected by the language changes. To timely react on unwanted results, the propagation of alerts should happen in real-time as soon as a change occurs. Modelling and domain experts can change models accordingly. The language engineers may oversee the changes. *Drawback:* changes are made by humans, which may be error-prone.
 - c. Keeping track of the language versions and correspondent models. This ability allows the user to trace back to the language versions that are consistent with the given models. *Drawback:* old models are not valid with the most recent version of a modelling language.

- d. Automated propagation of changes from the modelling language to the models. The human intervention is avoided. However, modellers and domain experts should still be able to make changes. *Drawback*: the automation is not feasible if modelling languages are not machine-interpretable.

Solution (a) prevents the language evolution during projects or after a language is operational. On the contrary, solutions (b), (c) and (d) allow perpetual changes to happen while projects are running or, in other words, while the language is in its full operation. Therefore, they are in line with the agile principles of the Business Agility Manifesto (Burlton et al., 2017). Hence, they can be considered as agile solutions.

The agile solutions (b) and (c) can be realised through technical implementations.

The agile solution (d) requires automation of the modelling language and models. As described in Section 2.12, automation can be supported by combining models or meta-models with machine-interpretable semantics. Such a combination, however, presents some challenges when performing on-the-fly domain-specific adaptations of modelling languages. While the challenges are discussed in the next section, the implementation of the solution (d) is left to future work.

- (3) The third issue arose when using the function create, i.e., *operators 1, 4 and 7*. The meaning of a new language construct (including their relations and attributes) may be ambiguous or unclear in models that are created with these language constructs. Ambiguous or unclear models can lead to erroneous interpretations, which can have severe consequences for the subsequent decision-making process. A modelling approach that makes the knowledge of modelling constructs explicit can help to reduce their ambiguity and unclarity. As stated by Battistutti and Bork (2017), it is not only a matter of making the knowledge explicit that is important, but also the form or representation of that knowledge. The form of knowledge can lead to an unambiguous interpretation of the meaning by others. Knowledge specified in natural language is considered informal, and when specified in a meta-model it is considered as semiformal. Such specifications leave room to subjective interpretations (Selic, 2007). A formal representation that specifies the meaning of a modelling element would remove its subjective interpretation. Additionally, Harel and Rumpe (2000) stress that the formality of the meaning depends on the formalism of the abstract syntax, the semantic domain and the semantic mapping. This formalism can be provided with an ontology language, which is machine interpretable. Hence, an approach that grounds modelling elements with a machine-interpretable semantics is a valid approach. This approach is further elaborated in the following section.

5.2 Towards a Machine-Interpretable Semantics for the Agile Meta-Modelling

A modelling language whose semantics are not machine-interpretable leads to several deficiencies in the context of the proposed agile meta-modelling approach:

1. Ambiguity of the meaning of modelling constructs. This issue is particularly relevant when an adaptation leads to create or update modelling constructs for which a clear meaning shall be shared among the stakeholders;
2. Semantic interoperability problems among existing modelling languages or DSMLs (Bräuer & Lochmann 2008; Parreiras 2012; Walter et al. 2014a). This issue is well-known in the practice of meta-modelling due to the low effort towards interoperability across the widely adopted meta-modelling tools (e.g. Eclipse Modelling Framework⁵⁰, ADOxx⁵¹, MetaEdit+⁵²) (Karagiannis et al., 2016).
3. The automation of modelling languages and models is not directly possible.

These three obstacles can be overcome by a semantic that is not only readable but also *machine-interpretable* (see also Section 2.12). A machine-interpretable semantics:

1. Supports a clear and unambiguous definition of the meaning of a modelling construct;
2. Provides semantic interoperability among modelling languages or DSMLs (see Bräuer and Lochmann, 2008; Höfferer, 2007);
3. Provides automation capabilities to the modelling language and the models built with it (Walter et al., 2014; Kritikos et al., 2018).

5.2.1 Semantic Lifting

A commonly adopted way to achieve machine-interpretable semantics is by establishing a mapping between the human-interpretable modelling construct and its related formal representation (e.g. through mathematical formulas, logic-based languages or ontologies). This creates a separation of models interpretable by humans, and knowledge interpretable by machines, which is elaborated in Section 2.12. For instance, Berardi et al. (2005) define the semantics of UML class diagrams by mapping the modelling constructs to description logics (Baader & Nutt, 2003), which is a fragment of first-order logic. The most widespread way of turning models into a machine-interpretable format is through the mapping of concepts with logic-based languages such as ontologies, i.e. semantic lifting or semantic annotation (see Section 2.12.4).

There is a distinction between the *semantic lifting* and the *semantic mapping* introduced in Section 5.1.4. The former aims to establish an ontological reflection (also called ontological meta-modelling) of concepts from a model, meta-model or higher abstraction (Höfferer, 2007). The semantic mapping, on the other hand, maps the abstract syntax into the semantic domain, where concepts of the abstract syntax are independent from concepts of the semantic domain (Harel & Rumpe, 2000). Hence, when applying the distinction proposed in (Atkinson & Kuhne, 2003) the ontological representation in the semantic lifting still corresponds to the *language definition*, whereas the semantic domain in the semantic mapping corresponds to the *domain definition*. Both are relevant when specifying a modelling language.

⁵⁰ <https://www.eclipse.org/modelling/emf/>

⁵¹ <https://www.adoxx.org/live/home>

⁵² <https://www.metacase.com/>

These notions are the theoretical foundations to design the ontology architecture for the proposed agile meta-modelling approach (see Section 5.3.2).

5.2.1.1 Inconsistency Issues of Semantic Lifting

As already stressed in Section 2.12.4, the main issue of the semantic lifting regards the consistency between the human-interpretable representation and the related machine-interpretable representation, which is represented in an ontology. Keeping them separate tends to cause incompatible semantics. Aligning meta-model concepts and ontology concepts (see upper part of Figure 103) is error-prone and time-consuming.

Mechanisms for the automatic transformation of models into ontologies partially relieve this effort (Emmenegger et al., 2017; Karagiannis & Buchmann, 2018). However, the consistency issue remains if a change occurs in the ontology, as the human-interpretable model has to be adapted accordingly. If changes occur in the meta-model, transformation patterns for the ontology generation should also be adapted (Emmenegger et al. 2016). The “input” arrow in Figure 103 depicts the need to consider the class structure of the ontology before performing the transformation of the models into ontology instances. Else, inconsistencies between the class structure and created instances arise.

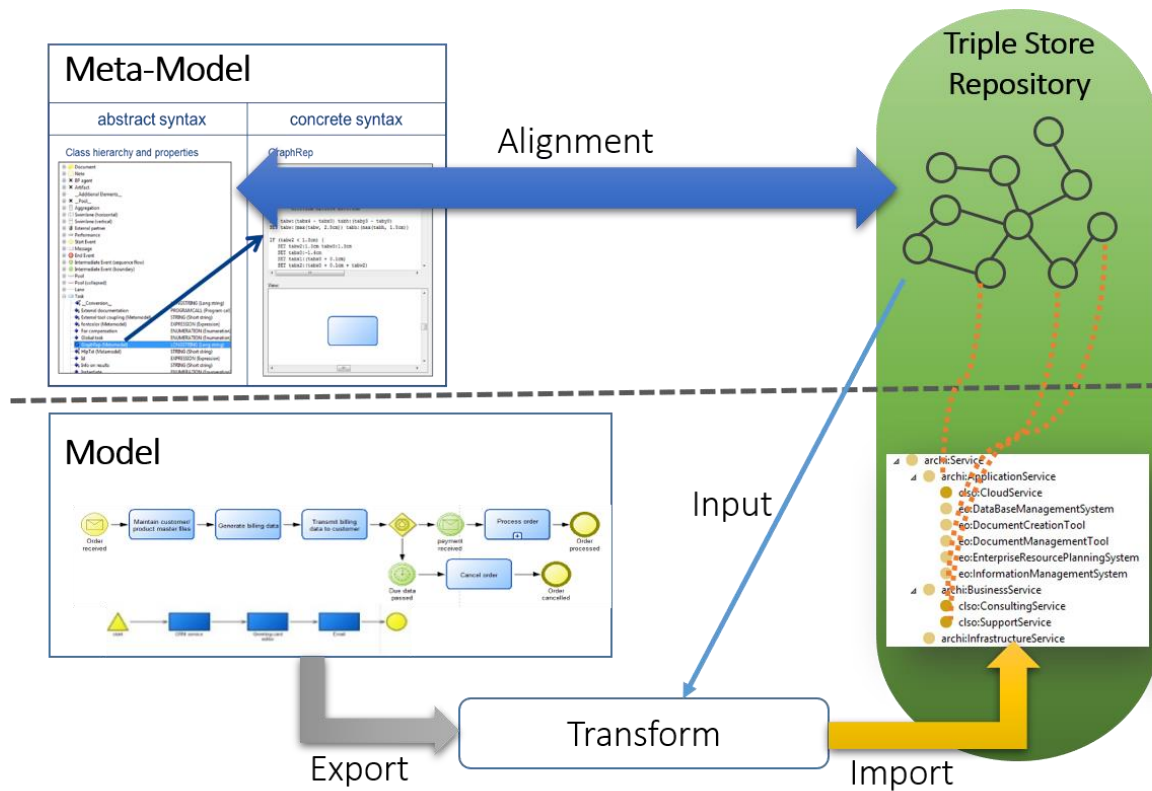


Figure 103. Alignment between meta-model and ontology; and transformation of models into ontology instances

5.2.2 Problems of Semantic Lifting in the Agile Meta-Modelling

Figure 104 shows the conceptualisation of an agile meta-modelling approach embracing the semantic lifting. Namely, the component that integrates language engineering and modelling is for the human interpretability (see right-hand side of Figure 104), while the ontology is for the machine interpretability (see left-hand side of Figure 104). Both abstract syntax and the

corresponding ontological representation refer to the linguistic definition (the semantic mapping and domain semantic specification are ignored as they are not relevant for the purpose of this sub-section).

Operators for the on-the-fly domain-specific adaptations are applied to affect modelling constructs in the human-interpretable part, i.e. abstract syntax and graphical notation (see right-hand side of Figure 104). The *meta*²-model of the abstract syntax contains concepts of the UML class diagram, whereas the *meta*²-model of the graphical notation contains specifications or graphical components that allows for drawing of the graphical notation. For instance, ADOxx implements the GraphRep functionality where a proprietary script is used to draw the graphical notation. The graphical notation is then associated to the class defined in the abstract syntax.

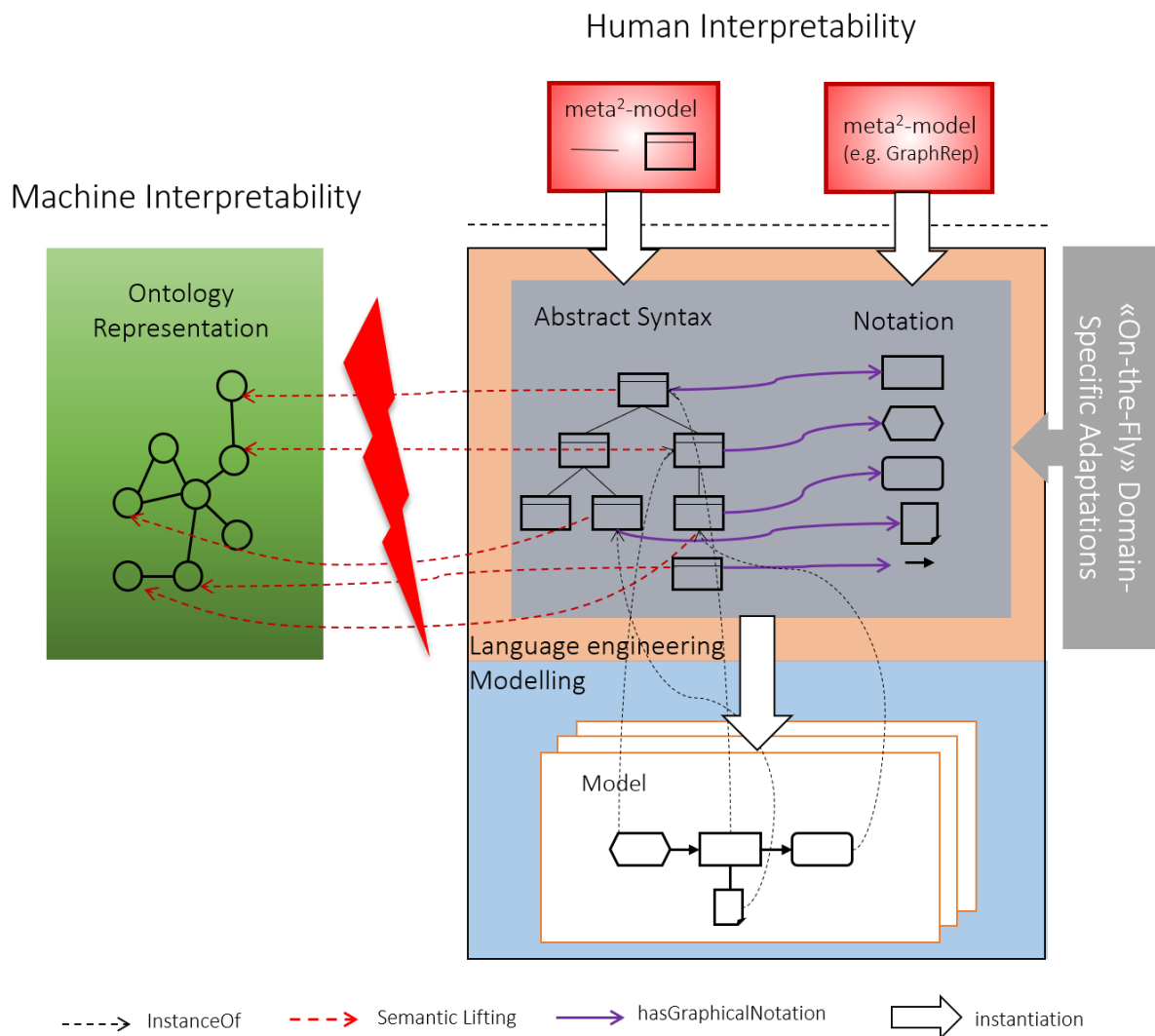


Figure 104. Inconsistency between the human- and machine-interpretable representations of modelling languages in the agile meta-modelling approach. Adapted from (Höfferer, 2007).

Let us assume that the required semantic lifting relationships are established between modelling constructs of an abstract syntax and the corresponding concepts in the ontology (see dashed red relations in the upper level of in Figure 104). on-the-fly domain-specific adaptations may raise the following drawbacks between the human- and machine representations:

1. Operators that apply the function Create lead to a lack of reflecting ontology concepts. For example, a newly created modelling construct in the abstract syntax would not be grounded with an ontology.
2. Operators that apply the function Update may raise inconsistencies with the corresponding ontology concept. For example, updating the name of a modelling construct in the abstract syntax will create an inconsistency with the name or label of the corresponding ontology concept;
3. Operators that apply the function Delete leads to mismatches between the human and machine interpretable representations. For example, an ontology concept that corresponds to a just removed modelling construct would remain in the ontology. This not only creates a mismatch between the machine and human knowledge representations, but it can also be problematic in the future when new ontology concepts are created, as they can create unwanted redundancies, e.g. if concepts are called differently but with the same meaning.

The red flash in Figure 104 symbolises the occurrence of drawbacks when performing on-the-fly domain-specific adaptations.

Each drawback can be overcome by adjusting the correspondent ontological representation. However, manual adjustments would not be convenient as they are time consuming and require both high maintenance effort and ontology expertise. Also, applying a transformation from the human- to the machine-interpretable representation each time an adaptation occurs is not an optimal solution.

This elaboration demonstrates the need to conceive an alternative approach, other than semantic lifting, to achieve an agile meta-modelling. The new approach should not only achieve the machine-interpretable of modelling languages but should also preserve model consistency when changes occur.

5.3 An Ontology-Aided Approach for the Agile Meta-Modelling

In order to support consistency between the human- and machine-interpretable representations for an agile meta-modelling, this section describes an ontology-aided approach. The aim to achieve a seamless alignment between the human- and machine-interpretable knowledge. A seamless alignment means that consistency between the two knowledge representations is preserved after changes occur in one or the other. The first prerequisite is to build upon an *ontology-based meta-modelling* (Sub-section 5.3.1), where changes that occur in the ontology are propagated to the graphical models. Then, Sub-section 5.3.2 introduces the *ontology architecture* that supports the proposed agile and ontology-aided meta-modelling approach. Next, mechanisms are introduced to aid the propagation of domain-specific adaptations from the graphical modelling language to its machine-interpretable representation (Sub-section 5.3.3).

5.3.1 The Ontology-based Meta-Modelling

The ontology-based meta-modelling approach introduced in (Hinkelmann, Laurenzi et al., 2018) merges the abstract syntax of meta-models with the corresponding ontological representation. The abstract syntax includes a semantic definition of the language but not the semantic domain.

In contrast to semantic lifting where UML class diagram concepts are used to specify the abstract syntax, in the presented approach the abstract syntax is specified by an ontology. In this sense, the ontology-based modelling can be regarded as a variant of the MOF meta-modelling framework (OMG, 2016c); where UML is replaced by an ontology language as a meta-modelling language, e.g. RDF(S). Ontology concepts are then associated with specifications of the graphical notation, e.g. a rectangle, square, gateway etc. The graphical notation for each ontology concept is defined separately from the semantic description (see separate *meta*²-models in Figure 105). The association is established through the definition of a mapping between the concept definition and the graphical definition (Nikles & Brander, 2009).

In the ontology-based meta-modelling approach, the semantics are expressed only once for both human-interpretable and machine-interpretable representations. Hence, the ongoing alignment between the modelling constructs of an abstract syntax and corresponding ontology concepts (see upper part of Figure 103) are not needed. The semantics in the ontology consists of classes, attributes, relations and constraints.

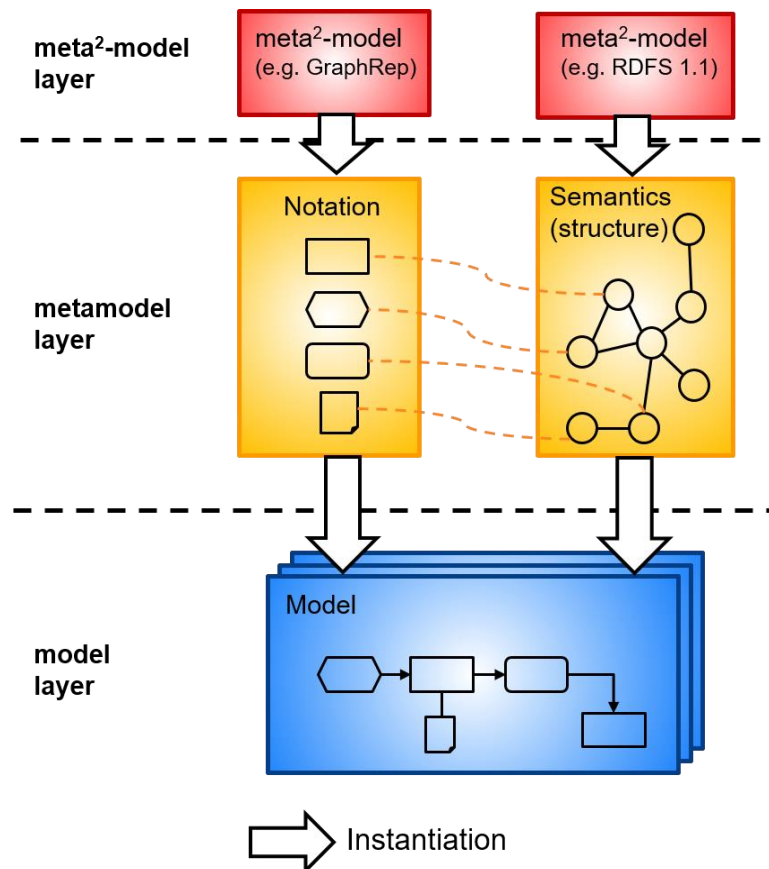


Figure 105. Ontology-based meta-modelling (Hinkelmann, Laurenzi et al. 2018)

In the ontology-based meta-modelling approach, the ontology itself is the meta-model for the graphical modelling environment (see model layer of Figure 105). A model is an instantiation of both ontology and related notations that reside in the meta-model layer. Thus, the model in the bottom layer benefits from both a semantics that is machine-interpretable, and a graphical notation, that makes it human-interpretable. Models are also ontology-based as they are instances of the class ontology. Such an approach avoids transforming the graphical models into ontology instances, in contrast to the approach described in Sub-section 5.2.1.1. Moreover, if changes occur in the ontology, these are propagated to the graphical models.

The validity of the approach was demonstrated by implementing it a context-adaptive questionnaire (Kritikos, Laurenzi, Hinkelmann 2018), where changes in the ontology directly led to changes in the questionnaire. Thus, the propagation of machine- to human-interpretable knowledge was ensured. Figure 106 shows the ontology-based meta-modelling for the context-adaptive questionnaire and corresponding question.

Ontology-based meta-model

- questionnaire:Question (26)
 - questionnaire:Availability (5)
 - questionnaire:Functional (3)
 - questionnaire:Interoperability (3)
 - questionnaire:Payment (1)
 - questionnaire:Performance (5)
 - questionnaire:Reliability (2)
 - questionnaire:Security (3)**
 - questionnaire:ServiceSupport (3)
 - questionnaire:TargetMarket (1)

[Resource]	rdfs:label
questiondata:Encryption_Type	What is your preferred data encryption level?
questiondata:What_is_your_favor...	What is y...
questiondata:Would_an_automat...	Do you n...

Graphical model

Figure 106. An ontology-based meta-modelling for a context-adaptive questionnaire

5.3.2 Ontology Architecture for an Agile and Ontology-Aided Meta-Modelling Approach

This sub-section elaborates on an ontology architecture that supports the proposed agile and ontology-aided meta-modelling approach. In particular, the architecture provides the basis to achieve a seamless consistency between the human and machine-interpretable knowledge of modelling languages. The seamless alignment is achieved not only by ensuring that changes are propagated from the machine- to the human-interpretable knowledge, but also vice-versa. In fact, the main changes come from the human-interpretable representation of the modelling language, through the on-the-fly domain-specific adaptations.

For the sake of clarity, from now on the ontology terminology is used to distinguish between attributes and relations, i.e. *datatype property* and *object property*, respectively. A datatype property refers to a relation between a subject resource and an object resource, where the object resource is a datatype, e.g. string, integer, boolean etc. whereas an object property refers to a relation between a subject resource and an object resource, where the object resource is a class or an individual (Allemang & Hendler, 2011).

The abstract syntax and semantics are specified by the RDF(S) ontology language and have a separate *meta*²-model for the graphical notation (see upper part of Figure 107). For the specification of the semantics we refer to Sub-section 5.1.4. The following notions are considered:

1. Graphical Notation,
2. Abstract syntax,
3. Semantic domain,
4. A mapping between concepts from the abstract syntax to those of the semantic domain.

Notions (1) and (2) can be found in (Frank 2013a). Notions (3) and (4) are introduced in (Harel & Rumpe, 2000) and Karagiannis and Kühn (2002), respectively. Additionally, the distinction between the abstract syntax and the semantic domain reverts to the distinction between language definition and the domain definition suggested in (Atkinson & Kuhne, 2003). As a result of these theoretical underpinnings, the following three main ontologies are conceptualised:

- Modelling Language Ontology (MLO),
- Domain Ontology (DO),
- Palette Ontology (PO).

The three ontologies are depicted in the ontology-based meta-model layer in Figure 107. The Modelling Language Ontology reflects the abstract syntax, while the Domain Ontology reflects the semantic domain. Concepts from the Modelling Language Ontology are mapped with concepts of the Domain Ontology (i.e. semantic mapping). Concepts in the Palette Ontology represent the graphical notations of the language for the palette, as well as for the graphical models; and are linked to the concepts of the Modelling Language Ontology. There are two different specifications for the graphical notation: one for the graphical models and one for the palette (see violet and yellow arrows pointing to two different notation types “for palette” and “for canvas” in Figure 107). This distinction allows showing different graphical notations between the palette and a graphical model. More details can be displayed in the graphical notation of a model element than the corresponding notation in the palette, where typically the space is limited.

A graphical model in the model layer is an instantiation of the ontology-based meta-model. Hence, the model elements and relations are ontology instances and thus remain machine-interpretable. In particular, a model element or relation (e.g., a concrete data object) is an instance of the following ontology concepts:

- a class in the Modelling Language Ontology (e.g. the class Data Object);
- an instance in the Palette Ontology (e.g. a Data Object instance containing the graphical notations);
- an instance or a class of the Domain Ontology (e.g. documentations or regulations within a specific domain of discourse).

Figure 107 depicts the three relations “*instanceOf*” starting from a model element and pointing to the concepts or instances of the three main ontologies; Modelling Language Ontology, Domain Ontology and Palette Ontology. The rationale is that the ontology instance reflecting the model element must inherit properties from classes, as well as from instances of the three main ontologies. This create the conditions for the modeller to:

- insert values for the properties that were created in the modelling construct,
- insert values for (or manipulate values from) the properties that exist in the domain ontology concept (or instance);
- manipulate property values of the Palette Ontology instances, i.e. changing the graphical notation that is visualised in the model, e.g. changing colours and size.

For this, an ontology language has to be able to support the *instance of an instance* abstraction representation, where the target instance can also be defined as a class. The choice of the ontology language is discussed after the next sub-section.

The ontology instances reflecting the graphical model are also instances of a class containing all the model instances. The class takes the name of the created model and belongs to the Ontology Model. This keeps modularity between the above seen three main ontologies and the ontology instances reflecting the models, thus ontology-based models can be exchanged as it is decoupled from the three main ontologies.

The model layer also contains the palette (see in the bottom-right corner of Figure 107.). The latter acts as a gate to the ontology-based meta-model enabling agility while modelling. For this, operators are applied on the palette allowing on-the-fly domain-specific adaptations of the meta-model layer.

The following sub-section provides a detailed description of the ontology-based meta-model specification.

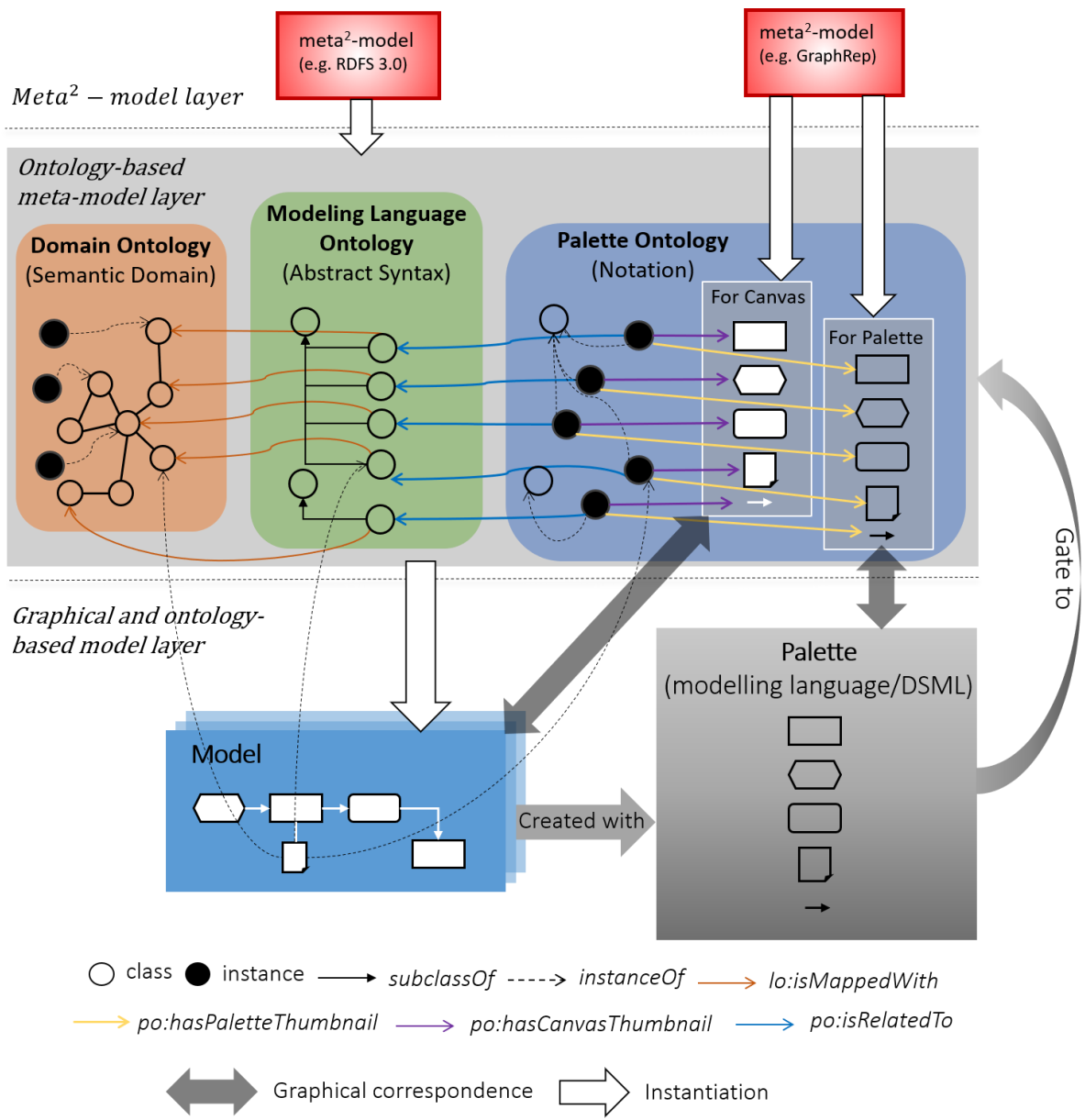


Figure 107. Architecture of the Ontology-Aided Approach

5.3.2.1 Specification of the Ontology-based Meta-Model

The description of the ontology-based meta-model is underpinned by Figure 108.

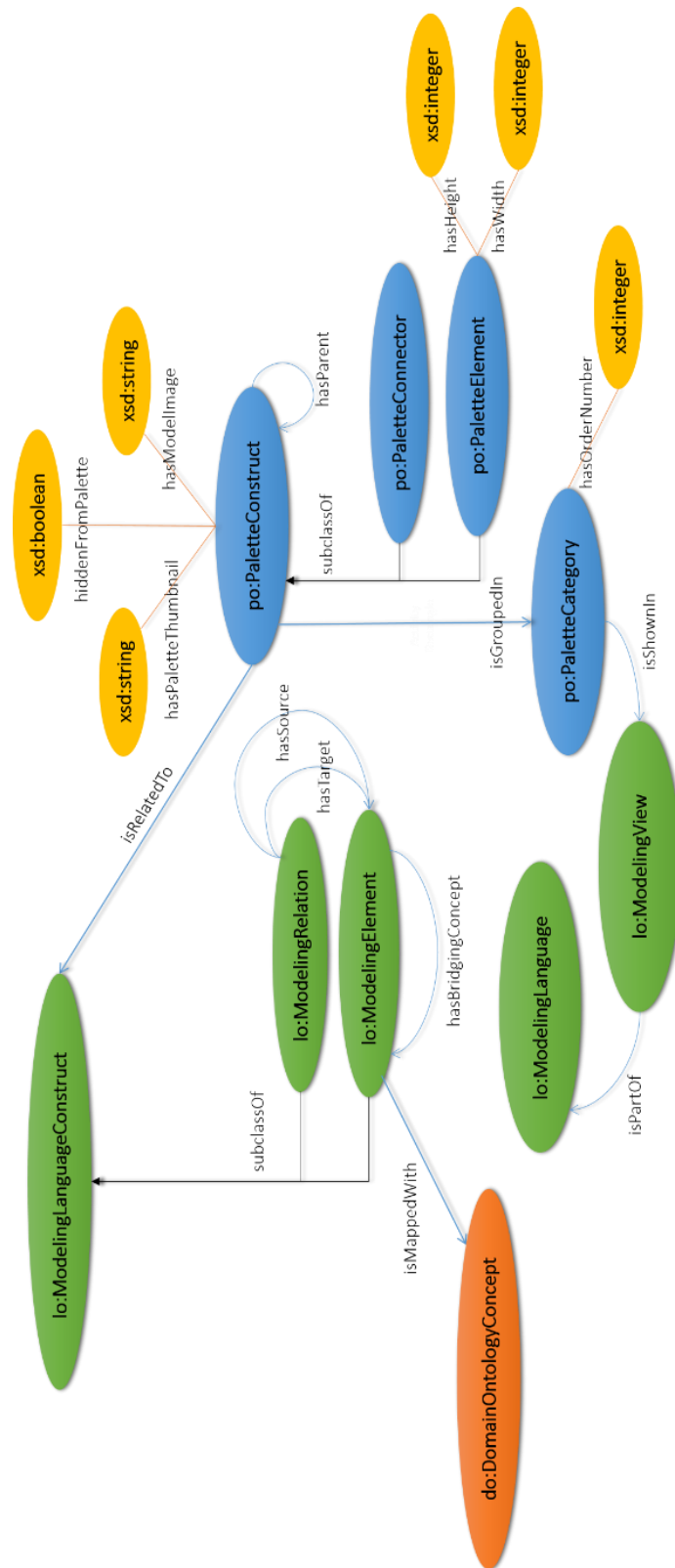


Figure 108. Ontology-based Meta-Model

The Palette Ontology (see blue bubbles in Figure 108) contains classes, object properties and instances that specify:

- the graphical notations (for palette and for the graphical model) of the modelling language (i.e. *po:PaletteConstruct*)
- knowledge for positioning the graphical notations over the palette (i.e. *po:PaletteCategory*).
- In detail, the class *po:PaletteConstruct* class has two sub-classes: *po:PaletteConnector* and *po:PaletteElement*.

po:PaletteConnector contains instances reflecting connectors of one or more modelling languages (e.g. message flow and sequence flow for BPMN), while the class *po:PaletteElement* contains instances reflecting modelling elements of one or more modelling languages (e.g. task or data object for BPMN). Instances from both classes inherit three datatype properties:

- *po:paletteConstructIsHiddenFromPalette*, which is associated with a Boolean datatype property. A “true” value means that the graphical notation for the palette will not be displayed in the palette, and a “false” value means that it will be displayed. The default value for each instantiation is “false”.
- *po:paletteConstructHasPaletteThumbnail*, which is associated with a string datatype property that will contain the Uniform Resource Identifier (URI) of a graphical notation to be shown in the palette;
- *po:paletteConstructHasModelImage*, which is associated with a string datatype property that will contain the Uniform Resource Identifier (URI) of a graphical notation to be shown in a graphical model;

The class *po:PaletteElement* has two datatype properties:

- *po:paletteConstructHasWidth*, which is associated with an integer datatype property that will contain the default value of the width of the graphical notation for the model;
- *po:paletteConstructHasHeight*, which is associated with an integer datatype property that will contain the default value of the height of the graphical notation for the model;

Note that the two datatype properties do not apply for modelling relations, thus they are inserted in the palette Element class. The two properties do not regard the size of the thumbnail to be displayed in the palette. The rationale is that all the thumbnails in the palette should have the same fixed width and height in order to have a homogenous appearance. Hence, they should be fixed values, which will be hardcoded in the user interface and will be used for any new thumbnail. Conversely, the image size to be shown in the model may vary from element to element. Although the two datatype properties are meant to store default values, they may be subject to change depending on the taste of the modeller or imposed modelling conventions, thus they should not be hard-coded.

Also, the class *po:PaletteConstruct* has five object properties:

- a self-relationship *po:paletteConstructHasParentPaletteConstruct*. This relation determines the hierarchy among modelling constructs with the purpose to show it in the palette.
- *po:paletteConstructIsRelatedToModellingConstruct* pointing to the class *lo:ModellingLanguageConstruct*. This object property connects instances of the classes *po:PaletteConnector* or *po:PaletteElement* (i.e. graphical notations) with classes of the

Modelling Language Ontology (i.e. abstract syntax). This relation reflects the link that connects notation with abstract syntax as described in (Karagiannis & Kühn, 2002).

- *po:paletteConstructIsGroupedInPaletteCategory* pointing to the class *po:PaletteCategory*. This object property specifies which Palette Constructs are grouped into which category.
- The class *po:PaletteCategory* has the purpose of grouping graphical notations of similar types into categories. When a modelling language or DSML presents many concepts that are to be shown in the palette, grouping them into categories is a good solution to avoid the cognitive overload. Among widely used modelling tools the grouping of the graphical notations varies. For instance, the on-line version of Camunda⁵³ and ADOxx keep separate connectors from modelling elements. The on-line version of Signavio⁵⁴ and Trisotech⁵⁵ show all the graphical notations in a single category. The modeller Bizagi⁵⁶ foresees several categories to group graphical notations in the palette, i.e. as Figure 109 shows, for BPMN the palette shows five categories: (1) flow elements, (2) connecting objects, (3) data (4) swimlanes, and (5) artifacts.

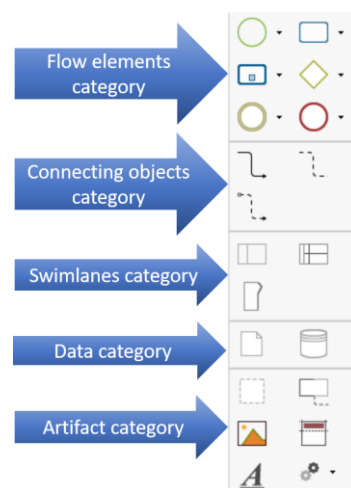


Figure 109. BPMN graphical notations grouped by categories in the Bizagi Modeller

The *po:PaletteCategory* has the following two datatype properties and one object property:

- *po:paletteCategoryHasOrderNumber*, which is associated to an integer datatype property that will contain a number. The number will be used for ordering the categories from top to bottom, where the value 1 corresponds to the top element in the palette;
- *po:categoryIsShownInModellingView*, which points to the class *lo:ModellingView*. This object property binds a category to a modelling view, so that for each selected modelling view the correspondent categories are shown in the palette. In turn, since each Palette Construct has its category, the latter is populated with the graphical notations of the constructs. The same category can be shared by two different modelling views. This is the case of the modelling language BPaaS (introduced in Section 4.2), which since extends BPMN, it also shares the same process modelling view.

⁵³ <https://demo.bpmn.io/new>

⁵⁴ <https://editor.signavio.com>

⁵⁵ <https://www.trisotech.com>

⁵⁶ <https://www.bizagi.com/en/products/bpm-suite/modeler>

The *Modelling Language Ontology* (see green bubbles in Figure 108) contains classes, taxonomy of classes, and properties (i.e. relations and attributes) describing the abstract syntax of a modelling language.

As shown in Figure 108, the Modelling Language Ontology is based on five classes:

- *lo:ModellingLanguage* – this class specifies the modelling language.
- *lo:ModellingView* – this class specifies the views of a modelling language. One or more modelling views comprise a modelling language. This knowledge is captured by the object property *isPartOf* between the two classes *lo:ModellingView* and *lo:ModellingLanguage*;
- *lo:ModellingLanguageConstruct* – this class generalises the concepts modelling elements and modelling relations.
- *lo:ModellingElement* – this class is a sub-class of *lo:ModellingLanguageConstruct* and specifies the modelling elements of a language. This class has two object properties: *isMappedWith* and *hasBridgingConcept*. The object property *isMappedWith* reflects the formal explication of the semantic mapping. This connects elements from the Modelling Language Ontology to those in the Domain Ontology. The self-relation of the Modelling Element class *hasBridgingConcept* indicates that a modelling element has a bridging concept targeting a modelling element from either the same modelling language (but different modelling view) or from a different modelling language. This relation allows the user to navigate between two elements when they are instantiated in the model.
- *lo:ModellingRelation* – this class is a sub-class of *lo:ModellingLanguageConstruct* and specifies the modelling relations of a modelling language. In a model, each instantiated relation has a source and a target modelling element. For example, a sequence activity in a BPMN model may have a start event as a source element and a user task as a target element. This knowledge representation is captured with the two object properties *hasSource* and *hasTarget*, which point to the class *lo:ModellingElement*;

The Modelling Language Ontology imports one or more modelling languages. Each Modelling Language Ontology concept has the prefix of the language it belongs to, e.g. Task in BPMN is shown as a class *bpmn:Task*.

The taxonomy of classes supports inheritance of properties from a class to its sub-classes. In ontology languages, like RDF(S), the inheritance mechanism is supported when specifying the *rdfs* property *subClassOf* between classes, where the sub-class inherits the properties specified in the class. This is convenient when extending a modelling construct as properties do not need to be re-created. Also, the inheritance mechanism applies in cascade from the first defined class down to the last sub-class. Therefore, when creating an instance of a class (i.e. a model element) all the properties that were defined in the class and the above super-classes are inherited, creating the conditions for a modeller to specify the properties.

The *Domain Ontology* (see orange bubble in Figure 108) contains classes and properties that describe the semantic domain. As mentioned above, the semantic domain is independent from the abstract syntax of a language and describes a domain of discourse. The Domain Ontology consist of existing ontologies that are imported to further specify a language construct. An example is the International Classification of Functioning, Disability and Health (ICF) ontology (National Center for Biomedical Ontology, 2012). The Domain Ontology can also be contextualised within the Linked Data (Auer, Berners-Lee, Bizer, & Heath, 2015) paradigm, which increasingly contain world-wide standards and vocabularies across the Internet. In this sense, a resource from the Domain Ontology can be linked with external

machine-interpretable resources. Such linkage enriches the context of a modelling construct, leading to a further specification of semantics. The *owl:sameAs* property can be established for resources having the same meaning.

Domain Ontologies are typically aligned with Upper Ontologies or also known as Top Level Ontologies, which contain a general semantic level, i.e. general terms like events, time, location. Examples of such ontologies can be found across the literature base (Pease et al. 2002; Hinkelmann et al. 2013; Chavula & Maria Keet 2015). Similar to the work in Emmenegger et al. (2013), where enterprise domain concepts are extended with general concepts, the Domain Ontology in this work extends to general and language-independent concepts like top level elements. Thus, both concepts from a Domain Ontology and concepts extended from the Domain Ontology can be mapped with concepts from the Modelling Language Ontology. The extension of the Domain Ontology with general concepts like top level elements is left to the ontology engineer as it does not concern the language engineering.

In order to support the interlinkage between the three ontologies in the ontology architecture, the Palette Ontology has to include the Modelling Language Ontology, which in turns has to include the Domain Ontology. Figure 110 shows such interlinkage, where each ontology contains the implemented ontologies. In the following sub-section the adoption of this ontology language is explained.

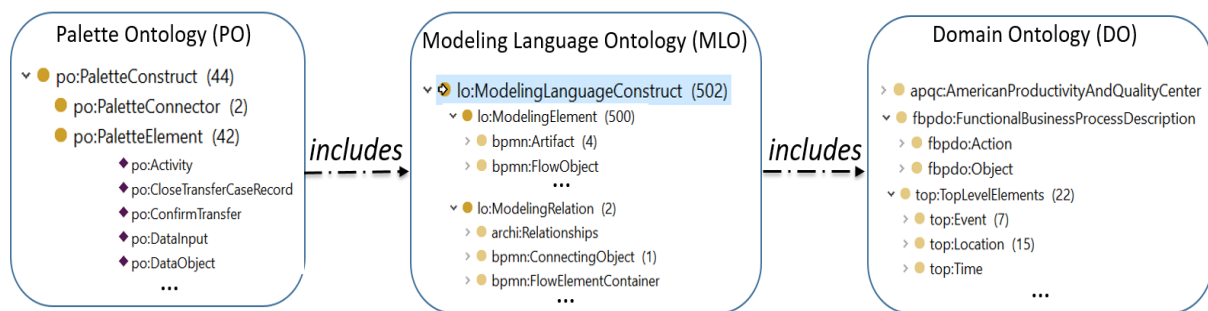


Figure 110. Implemented ontologies and their interlinkage

5.3.2.2 RDF(S) as Ontology Language

An introduction to ontology, its definition and most widely-adapted ontology languages are reported in Section 2.12 of the Literature Review chapter. Thus, the following description is limited to illustrate the choice of the language for the previously introduced ontology conceptualisation.

The choice of a ontology language typically depends on the purpose the ontology, i.e. types of facts that are important to deduce, represent and/or retrieve (Brachman, Levesque, & Pagnucco, 2004). In this work, the Resource Description Framework Schema (RDF(S)) 1.1 (W3C, 2014c) is adopted. This lightweight ontology language fits the proposed operators for the on-the-fly domain specific adaptations, e.g. create sub-classes, attributes, relations.

Moreover, it allows the use of classes as instances of other classes, on the contrary to the more expressive OWL (McGuinness & van Harmelen, 2014). OWL is limited to the knowledge representation of two levels: the TBox (i.e. classes) and the ABox (i.e. instances). This representation makes OWL unable to support a multi-layer representation that characterises meta-modelling representations (Fanesi, Cacciagrano, & Hinkelmann, 2015).

Instead, by adopting RDF(S) in the proposed agile meta-modelling approach, instances from the Palette Ontology can be further instantiated in the creation of models. Also, multilayer representation is supported. This would allow to model execution data as a further instance layer of models representing, for example, process activities. Semantic rules (e.g. SPIN (W3C, 2011)) and the SPARQL (W3C, 2008a) query language can be performed against ontologies expressed in RDF(S). SPIN or SPARQL CONSTRUCT are used to infer new knowledge while the SPARQL provides a powerful query construction. Several research works (Kritikos et al. 2018; Emmenegger et al. 2017; Emmenegger et al. 2013) show the validity of this approach.

Moreover, in contrast to more expressive ontology languages like OWL, RDF(S) embraces the *Closed World Assumption* (CWA) and satisfies the *Unique Names Assumption* (UNA). In CWA, it is assumed the complete knowledge of the world, therefore if something is not derived it is deduced that it does not exist. CWA is opposite than the *Open World Assumption* (OWA), where the knowledge of the world is assumed to be incomplete, hence if something is not derived it cannot be inferred that does not exist. The OWA becomes problematic for the validation of models (Rospocher, Ghidini & Serafini; 2014). For example, if a model has a gateway with an incoming sequence flow but no outgoing sequence flow, the model would not result logically inconsistent, which violates the semantics of BPMN.

The UNA, on the other hand, makes two entities with different identifiers distinct objects, which is a beneficial when creating models as each model construct is uniquely identified. More expressive ontology languages like OWL do not satisfy UNA, thus each time an element is entered in a model, new disjoint statements would have to be injected if the same entity type already exists in the model. Such work around adds complexity to the creation of models.

Although the model creation and validation are left to future work, it is appropriate to make such considerations at this stage as models inherit the ontology specification of the modelling languages. Therefore, the benefits of CWA and UNA on models is an additional argument to adopt RDF(S) as an ontology language

The terminology that will be adopted from now on belongs to the RDF(S) 1.1. In particular:

- the term *resource* refers to anything represented in an ontology or graph. *Rdfs:Resource* is the class of everything and all things described by RDF are resources;
- the term *class* refers to *rdfs:Class*, which declares a resource as a class for other resources;

- the term *property* refers to an instance of the class *rdf:Property* and describes a relation between a subject resource and an object resource, where property can also be named as *predicate*, *i.e.* the triple *subject-predicate-object*;
- the term *domain* refers to the *rdfs:domain* of an *rdf:Property* and declares the class of the *subject* resource in a triple. E.g. in the declaration “*pX rdfs:domain sY*”, the *sY* is the subject resource of the triple “*sY-pX-oZ*”;
- the term *range* refers to the *rdfs:Range* of an *rdf:Property* and declares the class or *datatype* of the *object* resource in a triple. E.g. in the declaration “*pX rdfs:range oZ*”, the *oZ* is the object resource of the triple “*sY-pX-oZ*”.

5.3.3 Designing Semantic Rules for the Propagation of Domain-Specific Adaptations

Sub-section 5.3.1 described the ontology-based meta-modelling, which allows the *propagation of machine- to human-interpretable knowledge*. The previous sub-section introduced the new ontology architecture to provide the basis for a seamless alignment between the human and machine-interpretable representation of modelling languages. This sub-section builds upon such an ontology architecture and proposes semantic rules for the *propagation of human- to machine-interpretable knowledge*. Semantic rules, therefore, aim to ensure that domain-specific adaptations of a modelling language are correctly propagated to the ontology. Such correct propagation enables the seamless or ongoing alignment between the human- and machine-interpretable representation of modelling languages.

In order to ensure the correct propagations of the adaptations, *11 SPARQL rules* are proposed, each of which implements one or more operators. A SPARQL rule consists of one or more triples in the form of *subject-predicate-object* and *insert, delete* or *update* knowledge in the ontology-based meta-model. Hence, each rule supports one or more of the 10 operators that were derived in Sub-section 5.3.1. As soon as a domain-specific adaptation is performed, the SPARQL rule of the related operator is instantiated. The instantiation contains the actual changes that need to be carried over to the ontology.

For example, let us consider the example depicted in Figure 111. The top of the figure shows the domain-specific adaptation that should be fulfilled, in this example the integration of two modelling elements from different modelling languages. The adaptation makes use of *Operator 1: Create sub-class*. The expected result is to create a sub-class relation between two modelling elements. To achieve that, the operator is supported by the two following SPARQL rules:

- Insert the relation “*hasParent*” between the two palette Element instances (in the Palette Ontology) that relate to the two modelling constructs to be integrated;
- Insert the relation “*subClassOf*” between the two modelling constructs (in the Modelling Language Ontology) to be integrated.

These SPARQL rules are, therefore, instantiated containing the actual instances of the palette element class and the two actual classes representing two modelling constructs. Thus, once rule instances are fired, the ontology-based meta-model is adapted with the new integration of the two modelling constructs.

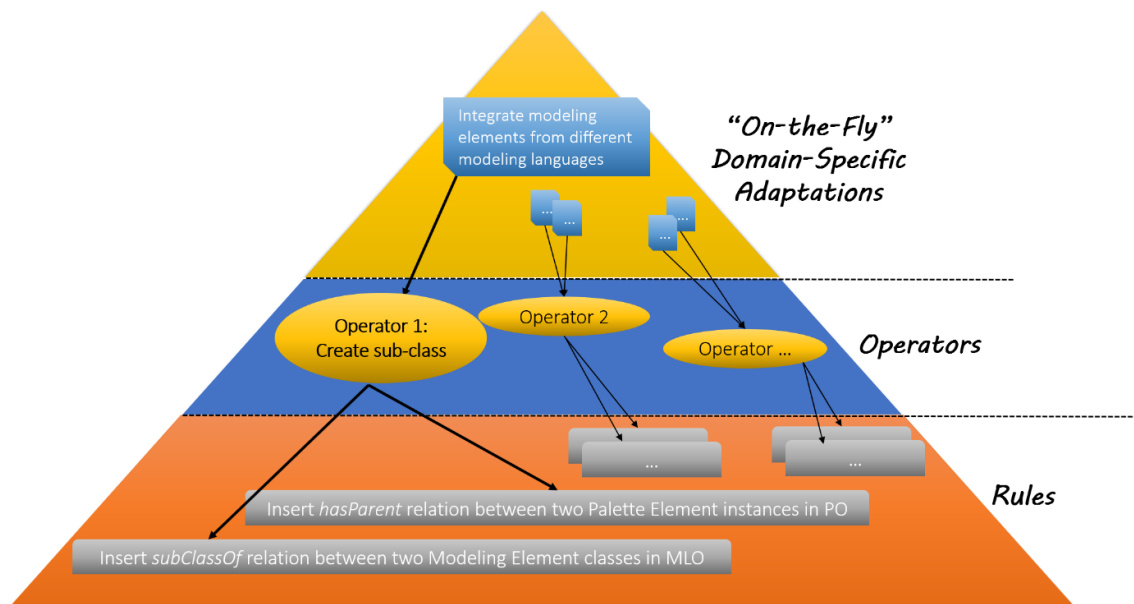


Figure 111. Semantic rules for the propagation of changes from the human to the machine-interpretable representation

5.3.3.1 Design and Evaluation of Semantic Rules

The design and evaluation of the semantic rules is supported by following the methodology proposed by Grüninger and Fox (1995). In the following, the general steps of the methodology are described. Then the manner in which the methodology was adopted for the development of semantic rules is discussed.

The methodology of Grüninger and Fox (1995) starts by explaining use case, from which informal competency questions (CQ) are derived and written in natural language. Then, concepts and relations are extracted from the questions to conceptualise an ontology. Next, the ontology and competency questions are formalised through an ontology language and a rule language, respectively. Lastly, the evaluation of the semantic queries or rules is performed by executing them with respect to the ontology. In result, it should be shown that the expected results are met. Hence, the ontology is tested by proving completeness theorems with respect to the competency questions. The methodology was successfully followed for the design and evaluation of several ontologies, e.g. the BPaaS Ontology (Woitsch et al., 2016), the ontology recommender for workplace learning (Emmenegger et al., 2017) and an ontology-supported procedure to assess procurement risks (Emmenegger et al., 2013).

The purpose of the Grüninger and Fox (1995) methodology in our case is not of designing an ontology, but support the rigorous development of semantic rules. The latter aims updating the ontology of the meta-model that already exists. Thus, some part of the methodology is slightly adapted to fit our purpose (see Figure 112). Namely:

- In the first step, domain-specific adaptations are described as a motivating scenario to design semantic rules.
- Next, informal rules (instead of competency question) are described in natural language.
- Finally, informal rules are transformed into SPARQL rules (instead of SPARQL queries).

The validation of the designed SPARQL rules is done with respect to its syntactic and semantic correctness.

- The syntactic correctness is validated by executing the SPARQL rules with the SPARQLer Update Validator⁵⁷.
- In order to ensure that the SPARQL rule produces the expected outcome, it is instantiated (see the SPARQL rule instances in Figure 112). This activity is underpinned by use cases from the patient transfer management case (Section 4.2). The chosen use cases represent typical domain-specific adaptations and cover all operators. The SPARQL rule instances are fired against the ontology-based meta-model⁵⁸. If results match with the expected ones it is the proof that a SPARQL rule is semantically valid.

SPARQL rules that are syntactically and semantically validated were considered for their implementation in the prototype described in Chapter 6. Namely, the validated SPARQL rules were embedded in Java methods for the dynamic generation of SPARQL rule instances.

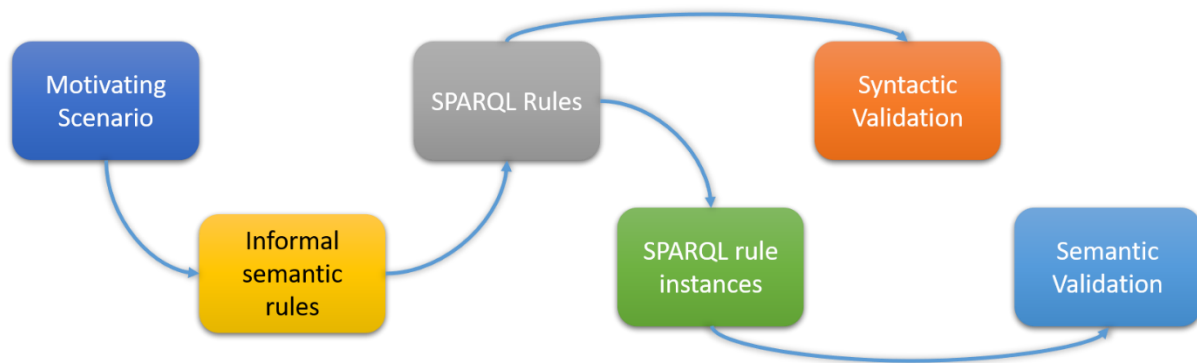


Figure 112. Methodology to design and evaluate semantic rules for the propagation domain-specific adaptations. Adapted from (Grüninger & Fox, 1995)

⁵⁷ <http://www.sparql.org/update-validator.html>

⁵⁸ The tests are done in the ontology editor TopBraid: <https://www.topquadrant.com/tools/ide-topbraid-composer-maestro-edition/>

5.3.3.2 Representation Language of the Semantic Rules

The SPARQL rules are represented in the W3C language specification SPARQL Update (W3C, 2008b). The SPARQL Update provides graph update operations such as “INSERT DATA” and “DELETE DATA” with which our 10 operators can be supported.

According to W3C (2008b) an operation is defined as “an action to be performed that results in the modification of data [...]” in a triplestore expressible as a single command, e.g. INSERT DATA or DELETE DATA. A triplestore is a mutable container of ontologies, which in our case reflect the afore-mentioned ontology-based meta-model. Operations on RDF(S) resources span classes, properties and instances. Namely, the “INSERT DATA” supports those operators that imply the creation of a resource, i.e. *Operator 1 – Create sub-class, Operator 4 – Create relation, Operator 7 – Create attribute and Operator 8 – Assign concept, attribute type or value*. The “DELETE DATA” statement supports *Operator 3: Delete sub-class, Operator 6: Delete relation, Operator 9: Delete Attribute*. The sequential use of both operations, first “DELETE DATA” and then “INSERT DATA”, supports those operators where an update is completed, i.e. *Operator 2: Update class, Operator 5: Update relation and Operator 10: Update attribute*.

5.3.4 Syntactic and Semantic Validation of Semantic Rules

All eleven semantic rules went through the steps of the methodology described in the previous sub-section. Hence, they are all syntactically and semantically validated. Both syntactic and semantic validation activities led to produce more than twenty screenshots, which were taken from the SPARQLer Update Validator and TopBraid, respectively. To avoid an unnecessary stretch of this thesis, the syntactic and semantic validation screenshots of only the first semantic rule (“*Integrating of Modelling Elements from different modelling languages*”) are reported in the text below. Therefore, the first rule is described with all the 6 steps of the above-mentioned methodology. The rest of the semantic rules (from 2 to 11) are then presented following the first four steps of the above-introduced methodology. Namely:

- (1) Description of one or more domain-specific adaptations.
- (2) Informal semantic rules that fulfil the domain specific adaptation(s) introduced in (1). For each rule, the supporting operator(s) is mentioned.
- (3) SPARQL specification for the semantic rule(s) introduced in (2).
- (4) Instances of the SPARQL rules that are specified in (3). Real-world scenarios from the patient transferal management case (Section 4.2) are considered for the instantiations of SPARQL rules.

All the screenshots for the syntactic and semantic validation of rules from 2 to 11 can be found in Appendix D: Validation SPARQL Rules, folders D1 and D2.

5.3.4.1 Rule 1: Modelling Elements Integration from different Modelling Languages

The first semantic rule aims to support the integration of two modelling elements from different modelling languages. Given two modelling constructs that belong to two different modelling languages, one modelling construct is conceptualised as an extension of the other. As an example, the integration of the Discretionary Task of CMMN is integrated into BPMN. It is conceptualised as an extension of the BPMN's Task element.

5.3.4.1.1 Informal Semantic Rule to Integrate Modelling Elements

The integration between modelling elements from different modelling languages requires a rule that creates the following two properties:

- Create one object property *hasParentPaletteConstruct* in the Palette Ontology between the instance of the class *PaletteElement* (*instanceSon*) and the instance of the class *PaletteElement* (*instanceParent*). *InstanceSon* and *instanceParent* relate to the modelling construct that is to be extended and the modelling construct that extends, respectively. Both modelling constructs are classes in the Modelling Language Ontology. The object property *hasParentPaletteConstruct* ensures a taxonomy among the instances that contain the graphical notations. The taxonomy is then graphically displayed in the palette.
- Next, create one property *subClassOf* in the Modelling Language Ontology between the modelling construct chosen for extending a class (*sourceClass*) and the one that is being extended (*targetClass*). This action creates the desired integration between the two modelling constructs in the Modelling Language Ontology. It also ensures that the class taxonomy in the Modelling Language Ontology is consistent with the above-mentioned graphically displayed taxonomy of the notation.

This rule supports Operator 1 – Create sub-class.

5.3.4.1.2 SPARQL Rule 1 to Integrate Modelling Elements

The informal rule for modelling elements integration is implemented by the SPARQL operation “INSERT DATA”. The latter fits the purpose of inserting one property *po:hasParentPaletteConstruct* between two instances and one property *rdfs:subClassOf*, between two classes.

SPARQL Rule 1 is shown in Table 30 and contains:

- the prefixes *rdfs*, *po* and *lo* that are used in the two statements. The prefix *rdfs* refers to the syntax of the RDF(S) ontology language. The prefix *po* refers to the Palette Ontology and *lo* to the Modelling Language Ontology.
- the operation “INSERT DATA”, the two statements that allow the creation of the two properties. The two properties are shown in bold.
- two comments above each statement (each comment is written in italics and starts with //).

All the variables written in italics *po:InstanceParent*, *po:InstanceSon*, *lo:sourceClass* and *lo:targetClass* will be replaced with concrete resources when the SPARQL Rule 1 is instantiated.

The rest of the SPARQL rules are shown with the same look as in Table 30.

Table 30. SPARQL Rule 1 – Integrate modelling elements from different modelling languages

```
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
PREFIX po:<http://fhnw.ch/modellingEnvironment/PaletteOntology#>
PREFIX lo:< http://fhnw.ch/modellingEnvironment/LanguageOntology#>
INSERT DATA {
//enter parent relation between two palette Element instances (triple 1)
po:InstanceSon po:paletteConstructHasParentPaletteConstruct po:InstanceParent .
//enter subClassOf relation between two Modelling Element classes (triple 2)
lo:sourceClass rdfs:subClassOf lo:targetClass .
}
```

5.3.4.1.3 Syntactic Validation of SPARQL Rule 1

As described at the beginning of the sub-section, the syntactic validation of the SPARQL rules were performed through the SPARQLer Update Validator. Figure 113 shows the screenshot that proves the syntactic validation of SPARQL Rule 1. Namely, the upper part of the figure contains the SPARQL rule described in Table 30, while the bottom of the figure shows the output. The output does not contain any errors, which means that SPARQL Rule 1 is syntactically correct.

As already mentioned, the screenshots for the syntactic validation of the rest SPARQL rules can be found in Appendix D: Validation SPARQL Rules, folder D1.

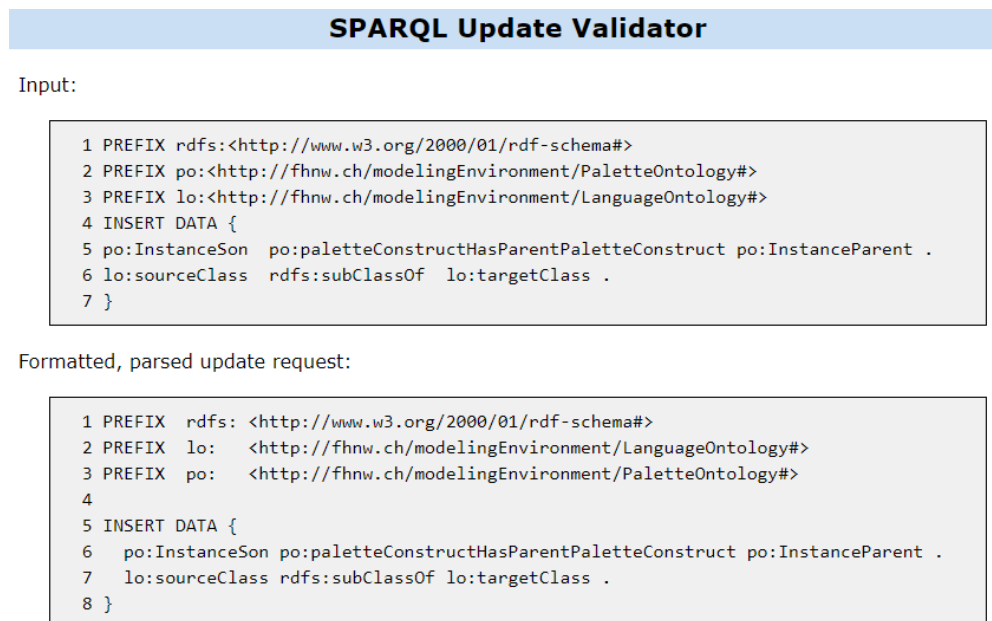


Figure 113. Syntactic Validation of SPARQL Rule 1

5.3.4.1.4 Instantiation of SPARQL Rule 1

In order to instantiate SPARQL Rule 1, the use case concerning the integration between the BPMN Manual Task and the CMMN Discretionary Task is considered. That is, the Discretionary Task is added as a sub-class of the Manual Task. Figure 114 shows the conceptualisation of this use case. The two arrows in Figure 114 indicate the two properties that shall be added by the two statements (or triple) of the SPARQL rule instance. These properties are the *subClassOf* relation between the Discretionary Task and the Manual Task and the *hasParent* object property between the two instances of *po:PaletteElement*.

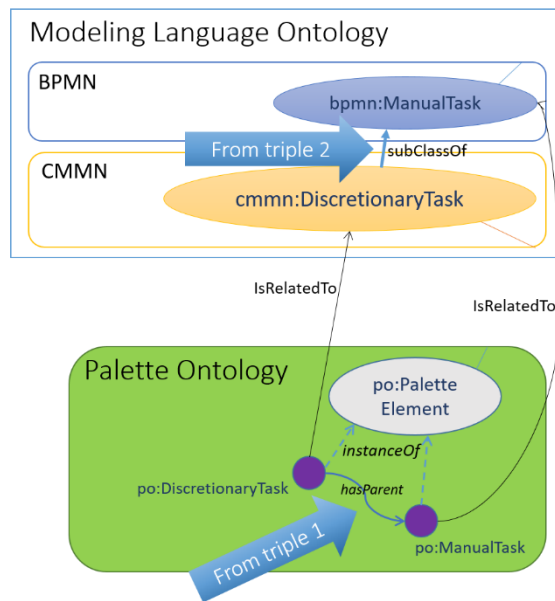


Figure 114 Conceptualisation of the use case for the integration of CMMN Discretionary Task with BPMN Manual Task

The instance of SPARQL Rule 1 is described in Table 31. The prefixes *bpmn* and *cmmn* on top of the rule instance replace the generic *lo* prefix seen in SPARQL Rule 1 (Table 30). Both prefixes refer to two ontologies, reflecting the linguistic view of the BPMN and CMMN modelling languages, respectively.

Table 31 An instance of SPARQL Rule 1

```

PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
PREFIX po:<http://fhnw.ch/modellingEnvironment/PaletteOntology#>
PREFIX bpmn:< http://ikm-group.ch/archimeo/BPMN#>
PREFIX cmmn:< http://ikm-group.ch/archimeo/CMMN#>
INSERT DATA {
//enter parent relation between the two palette Element instances (triple 1)
po:DiscretionaryTask po:hasParentPaletteConstruct po:ManualTask .
//enter subClassOf relation between the two Modelling Element classes (triple 2)
cmmn:DiscretionaryTask rdfs:subClassOf bpmn:ManualTask .
}

```

5.3.4.1.5 Semantic Validation of SPARQL Rule 1

In order to validate the semantic correctness of SPARQL Rule 1, its instance in Table 31 was fired against the ontology. For this, the ontology editor TopBraid was used. Figure 115 shows the result of the rule instance execution. As shown by the two arrows, the two triples contain the two new properties that have been added to the ontology. Hence, the Discretionary Task and the Manual Task are integrated as expected (see use case in Figure 114), which proves that SPARQL Rule 1 is correct not only syntactically, but also semantically.

As previously stated, the screenshots for the semantic validation of the rest SPARQL rules can be found in Appendix D: Validation SPARQL Rules, folder D2.

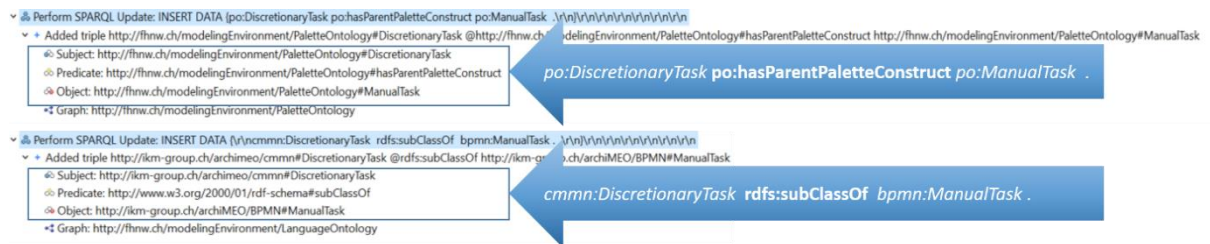


Figure 115. Semantic validation of SPARQL Rule 1 - modelling elements integration

5.3.4.2 Rules 2 to 5: Modelling Language Extension with a new Modelling Elements

The semantic rules from 2 to 5 aim to support the extension of a modelling construct with a new one. The extension includes not only a new modelling construct, but may also include:

- the creation of object properties for both the semantic mapping and the bridging connector of the new modelling construct;
- the creation of new datatype properties (including the assignment to a value type) for the new modelling construct;
- the creation of a new domain ontology concept.

Respectively, such domain specific adaptations are supported by four semantic rules:

1. one rule creates new modelling constructs,
2. one rule creates new object properties,
3. one rule creates new datatype properties,
4. one rule creates new domain concepts.

5.3.4.2.1 Informal Semantic Rule 2 to Create Modelling Constructs

The informal semantic rule for the new modelling constructs creates:

- a new class (*NewClass*) and a new relation *subClassOf* in the Modelling Language Ontology. *NewClass* is sub-class of the modelling construct that is extended (*ExtendedClass*). The creation of the *NewClass* includes the association of a label (*nameOfTheClass*) and may also contain any potential descriptive or informative comments (*descriptionOfNewConcept*).
- a new instance (*instanceOfNewClass*) in the Palette Ontology, containing *nameOfTheClass*, which is the same label of *NewClass*.

This rule supports Operator 1 – Create sub-class and Operator 8 – Assign attribute type or value.

Next, the informal semantic rule assigns the following *predicate-object* pairs to *instanceOfNewClass*:

- *PaletteConstructIsRelatedToModellingConstruct*, which points to the *NewClass*;
- *PaletteConstructHasParentPaletteConstruct*, which points to the parent instance of *instanceOfNewClass*. The parent instance already has a relationship with *ExtendedClass*;
- *PaletteConstructHasPaletteThumbnail*, which points to a string datatype. The datatype value will then be the file name of the chosen graphical notation that will be displayed in the palette;
- *PaletteConstructHasModellImage*, which points to a string datatype. The datatype value will then be the file name of the chosen graphical notation that will be displayed in the model;
- *PaletteConstructHiddenFromPalette*, which points a Boolean datatype with a default value set to “false”.

These additional five properties support Operator 8 – Assign concept, attribute type or value.

The first two properties are object properties, whereas the rest are datatype properties. All of the five properties are inherited from the class palette *Element* and are part of the schema of the Palette Ontology (see Sub-section 5.3.2).

5.3.4.2.2 Informal Semantic Rule 3 to Create Object Properties

This informal semantic rule creates a new object property for the new modelling construct. The object property can either reflect a semantic mapping (*isMappedWith*) or a bridging connector (*hasBridgingConcept*). Thus, these object properties have the new modelling construct as source and the selected class as target. The selected class can be a domain ontology concept of a modelling element, respectively. If a new object property is a semantic mapping it should be added as a sub-property of it. The same applies for new properties of the bridging connectors.

This rule is omitted if there are no semantic concepts nor bridging concepts selected by the language engineer.

This rule supports Operator 4 – Create relation.

5.3.4.2.3 Informal Semantic Rule 4 to Create Datatype Properties

This informal semantic rule creates new datatype properties for the new modelling construct. The possible built-in datatypes to be chosen for the given datatype property are the RDF-compatible XSD types and conform to the XML Schema built-in datatypes, i.e. Boolean, Date, DateTime, Decimal, Integer, String etc. (see in (W3C, 2014c)). In case datatype properties are not entered, this semantic statement is not generated.

This rule is omitted if there are no new datatype properties entered by the language engineer.

This rule supports Operator 7 – Create attribute and Operator 8 – Assign concept, attribute type or value.

5.3.4.2.4 Informal Semantic Rule 5 to Create Domain Concepts

This informal semantic rule creates new sub-classes in the Domain Ontology. Namely, the rule creates a new relation *subClassOf*, which points to the domain concept being extended.

Next, a label for the new domain concept is also created.

This rule is omitted if no new domain concept is entered by the language engineer.

This rule supports Operator 1 – Create sub-class.

5.3.4.2.5 SPARQL Rule 2 to Create Modelling Constructs

The informal rule 2 is transformed into a SPARQL rule, which turns into the SPARQL Rule 2 presented in Table 32. Like in the SPARQL Rule 1, comments are reported above each triple and all resource names are written in italics (e.g. *lo:NewClass*, *lo:ExtendedClass*, etc.) and are replaced with concrete resources when the SPARQL rule is instantiated.

Table 32. SPARQL Rule 2 – Create modelling construct

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX lo: <http://fhnw.ch/modellingEnvironment/LanguageOntology#>
PREFIX po: <http://fhnw.ch/modellingEnvironment/PaletteOntology#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
INSERT DATA {
//creation of the new class, subclass property and label in the Modelling Language Ontology
lo:NewClass rdfs:type rdfs:Class .
lo:NewClass rdfs:subClassOf lo:ExtendedClass .
lo:NewClass rdfs:label "name of the new class".
lo:NewClass rdfs:comment "comment for the new class".
//creation of the new instance in the palette Ontology
po:InstanceOfNewClass rdfs:type po:PaletteElement .
//association of property-object pairs to the new instance
po:InstanceOfNewClass rdfs:label "same name entered for the new class".
po:InstanceOfNewClass po:paletteConstructIsRelatedToModellingLanguageConstruct
lo:NewClass .
po:InstanceOfNewClass po:paletteConstructHasParentPaletteConstruct
po:InstanceOfExtendedClass .
po:InstanceOfNewClass po:paletteConstructHasPaletteThumbnail "name of
image"^^xsd:string .
po:InstanceOfNewClass po:paletteConstructHasModelImage "name of image"^^xsd:string .
po:InstanceOfNewClass po:paletteConstructHasWidth "number of pixel"^^xsd:integer .
po:InstanceOfNewClass po:paletteConstructHasHeight "number of pixel"^^xsd:integer .
po:InstanceOfNewClass po:paletteConstructIsGroupedInPaletteCategory lo:PaletteCategory .
//make the palette thumbnail visible in the palette
po:InstanceOfNewClass po:paletteConstructIsHiddenFromPalette "false"^^xsd:boolean .
}
```

5.3.4.2.6 SPARQL Rule 3 to Create Object Properties

The informal rule 3 is transformed into the SPARQL Rule 3 and shown in Table 33. As already described in the information rule 2, there are two kind of object properties that are created: one for the semantic mapping (see rule A in Table 33) and one for bridging connector (see rule B in Table 33). The SPARQL Rule 3a shows the creation of a new semantic mapping between a modelling element (*lo:NewClass*) and a domain concept (*do:ClassDO1*). The new semantic mapping is sub-property of *lo:elementIsMappingWithDOConcept*, which is the name of the main property connecting the class *lo:ModellingElement* with *do:DomainConcept* (see Sub-section 5.3.2.1). The SPARQL Rule 3b creates a new bridging connector between a modelling element (*lo:NewClass*) and another modelling element *lo:ClassMLO1*. The new bridging connector is a sub-property of *lo:elementHasBridgingConcept*. Labels are also created for each object property. The prefixes shown on the upper part of Table 33 also apply for the rule that creates bridging connector.

Table 33. SPARQL Rule 3 – Create object property

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
PREFIX lo:<http://fhnw.ch/modellingEnvironment/LanguageOntology#>
PREFIX do: <http://fhnw.ch/modellingEnvironment/DomainOntology#>

(3a)
INSERT DATA {
//Creation of a new object property: a semantic mapping relation between a Modelling Language
Ontology class and a Domain Ontology class.
lo:isMappedWithClassDO1 rdf:type rdf:Property .
lo:isMappedWithClassDO1 rdfs:label "New class connected with class DO1" .
lo:isMappedWithClassDO1 rdfs:domain lo:NewClass .
lo:isMappedWithClassDO1 rdfs:range do:ClassDO1 .
lo:isMappedWithClassDO1 rdfs:subPropertyOf lo:elementIsMappedWithDOConcept .
}

(3b)
INSERT DATA {
//Creation of a new object property: a bridging connector relation between concepts in the Modelling
Language Ontology
lo:hasBridgingConceptMLO1 rdf:type rdf:Property .
lo:hasBridgingConceptMLO1 rdfs:label "New class connected with class MLO1" .
lo:hasBridgingConceptMLO1 rdfs:domain lo:NewClass .
lo:hasBridgingConceptMLO1 rdfs:range lo:ClassMLO1 .
lo:hasBridgingConceptMLO1 rdfs:subPropertyOf lo:elementHasBridgingConcept.
}
}
```

5.3.4.2.7 SPARQL Rule 4 to Create Datatype Properties

The informal rule 4 is transformed into SPARQL Rule 4, which is shown in Table 34. SPARQL Rule 4 creates new datatype properties (e.g. *hasAttrF* and *hasAttrE*) for the new modelling construct (*lo:NewClass*). Labels are also created for each datatype property. To show an example of possible in-built datatypes, the properties *hasAttrF* and *hasAttrE* are assigned to *xsd:String* and *xsd:Boolean*, respectively.

Table 34. SPARQL Rule 4 – Create datatype property

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX lo: <http://fhnw.ch/modellingEnvironment/LanguageOntology#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
INSERT DATA {
// creation of new data type properties
lo:hasAttrF rdf:type rdf:Property .
lo:hasAttrE rdf:type rdf:Property .
lo:hasAttrE rdfs:label "New class has attribute E".
lo:hasAttrF rdfs:label "New class has attribute F".
lo:hasAttrF rdfs:domain lo:NewClass .
lo:hasAttrE rdfs:domain lo:NewClass .
lo:hasAttrF rdfs:range xsd:string .
lo:hasAttrE rdfs:range xsd:boolean .
}
```

5.3.4.2.8 SPARQL Rule 5 to Create Domain Concepts

The informal semantic rule 5 is transformed into SPARQL Rule 5, which is reported in Table 35. The SPARQL rule creates new domain concepts (*do:NewDOconcept*) and labels. The property *rdfs:subClassOf* is also added between the new domain concept and the one being extended.

Table 35. SPARQL Rule 5 – Create domain concept

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX do: <http://fhnw.ch/modellingEnvironment/DomainOntology#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
INSERT DATA {
// creation of new domain ontology concept
do:NewDOconcept rdf:type rdfs:Class .
do:NewDOconcept rdf:label "New class in DO" .
do:NewDOconcept rdfs:subClassOf do:ClassInDO .
}
```

5.3.4.2.9 Instantiation of SPARQL Rules 2 to 5

In order to instantiate SPARQL rules from 2 to 5 a real-world use case from the Patient Transferal Management case is considered:

A concept from the Document and Knowledge Meta-Model is extended to conceptualise a document reflecting the International Classification of Functioning, Disability and Health (ICF) standard (World Health Organisation, 2016) (see also Sub-section 4.1.4). Figure 116 depicts the “as-is” situation, where the Modelling Language Ontology contains the ontologies reflecting the Document and Knowledge Meta-Model and BPMN. The concepts of interest in this scenario are the class *dkmm:Data_Document* and the *bpmn:DataObject* (see upper part of Figure 116). It is assumed that *bpmn:DataObject* was already extended with the new concept called *dsml4ptm:KoGuDataObject*, which contains all relevant information for the cost reimbursement. The Palette Ontology contains the graphical notations related to the introduced concepts (see bottom of Figure 116). To keep the use case simple, each instance in the Palette Ontology refers to one graphical notation for both the palette and the models.

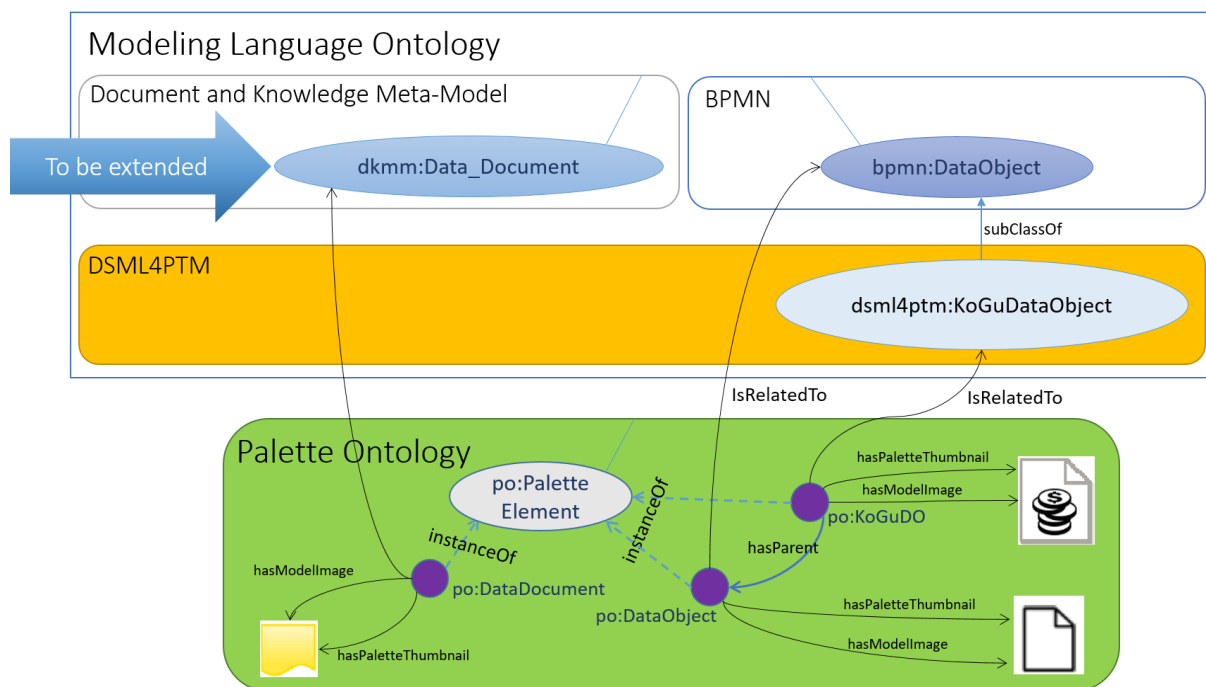


Figure 116. “As-is” use case for the extension of the *Data_Document* concept in the Document and Knowledge Meta-Model

In this scenario the language engineer starts by specifying the class *dkmm:Data_Document* with a new class *dsml4ptm:ICFStandard*. For this, SPARQL Rule 2 is instantiated, which is shown in Table 36. In detail, the SPARQL rule 2 instance creates a new class in the Modelling Language Ontology and the related instance in the Palette Ontology. Additionally, property-value pairs are associated to the instance such as label, comment, graphical notations and the property to display it in the palette.

Table 36. Instance of SPARQL Rule 2

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dsml4ptm: <http://fhnw.ch/modellingEnvironment/DSML4PTM#>
PREFIX dkmm: <http://fhnw.ch/modellingEnvironment/dkmm#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX po: <http://fhnw.ch/modellingEnvironment/PaletteOntology#>
INSERT DATA {
//creation of the new sub-class in the Modelling Language Ontology
dsml4ptm:ICFStandard rdf:type rdfs:Class .
dsml4ptm:ICFStandard rdfs:subClassOf dkmm:Data_Document .
//association of the new class with label, comment (annotation properties)
dsml4ptm:ICFStandard rdfs:label "ICF Standard".
dsml4ptm:ICFStandard rdfs:comment "This concept describes...".
//creation of the new instance in the palette Ontology
po:ICFStandard rdf:type po:PaletteElement .
//association of label to the new instance
po:ICFStandard rdfs:label "ICF Standard" .
//association of the instance with the related modelling construct
po:ICFStandard po:paletteConstructIsRelatedToModellingConstruct dsml4ptm:ICFStandard .
//association of the new instance with the instance that related with the parent-class of the new class
po:ICFStandard po:paletteConstructHasParentPaletteConstruct po:DataDocument .
//association of the new instance with the graphical notation palette thumbnail to be shown in the
palette
po:ICFStandard po:paletteConstructHasPaletteThumbnail
"gnICFStandardForPalette.png"^^xsd:string .
//association of the new instance with the graphical notation model image to be shown in the model
po:ICFStandard po:paletteConstructHasModelImage "gnICFStandardForCanvas.png"^^xsd:string .
//association of the new instance with the width of the graphical notation to be shown in the model
po:ICFStandard po:paletteConstructHasWidth "70"^^xsd:integer .
//association of the new instance with the height of the graphical notation to be shown in the model
po:ICFStandard po:paletteConstructHasHeight "100"^^xsd:integer .
//association of the new instance with the palette category
po:InstanceOfNewClass po:paletteConstructIsGroupedInPaletteCategory lo:PaletteCategory .
//make the palette thumbnail visible in the palette
po:InstanceOfNewClass po:paletteConstructIsHiddenFromPalette "false"^^xsd:boolean .
}

```

An additional requirement has to be accommodated: the ICF standard document must include the date and time of when it is created. Hence, the language engineer enters a new datatype property to the class *dsml4ptm:ICFStandard* with the in-built datatype *xsd:DateTime*. For this the SPARQL Rule 4 is instantiated (see Table 37). The rule instance creates a new datatype property *dsml4ptm:ICFStandardHasTimeStamp* with the class *dsml4ptm:ICFStandard* as domain and the *xsd:DateTime* as range.

Table 37. Instance of SPARQL Rule 4

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dsml4ptm: <http://fhnw.ch/modellingEnvironment/DSML4PTM#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
INSERT DATA {
//creation of the new datatype property "hasTimeStamp"
dsml4ptm:ICFStandardHasTimeStamp rdf:type rdf:Property .
dsml4ptm:ICFStandardHasTimeStamp rdfs:label "ICF Standard is assigned at" .
dsml4ptm:ICFStandardHasTimeStamp rdfs:domain dsml4ptm:ICFStandard .
dsml4ptm:ICFStandardHasTimeStamp rdfs:range xsd:dateTime .
}

```


The cost reimbursement for each patient’s case must include the patient’s disability specifications, which must be conform to the ICF Standard. In response, the language engineer is required to:

- Conceptualises the new class *dsml4ptm:ICFStandard* as part of the class *dsml4ptm:KoGuDataObject*. This conceptualisation allows modelling where a patient’s case cost reimbursement is linked to a document containing the patient’s case ICF specifications. For this, SPARQL Rule 3b is instantiated (see Table 39). The rule instance creates the bridging connector *isPartOf* (object property), which points to the class *dsml4ptm:KoGuDataObject*. Once instantiated in a model, the bridging connector will then allow navigating between instances *dsml4ptm:ICFStandard* and *dsml4ptm:KoGuDataObject*.

Table 38. Instance of SPARQL Rule 3b

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
PREFIX dsml4ptm: <http://fhnw.ch/modellingEnvironment/DSML4PTM#>
INSERT DATA {
//creation of the new object property “isPartOf” (bridging connector)
dsml4ptm:isPartOfKoGuDO rdf:type rdf:Property .
dsml4ptm:isPartOfKoGuDO rdfs:domain dsml4ptm:ICFStandard .
dsml4ptm:isPartOfKoGuDO rdfs:range dsml4ptm:KoGuDataObject .
dsml4ptm:isPartOfKoGuDO rdfs:subPropertyOf lo:hasBridgingConcept .
}

```

- Specify the modelling element *dsml4ptm:ICFStandard* with concepts of the already existing ICF Ontology. The latter reflects the class hierarchy of the whole ICF Standard and is issued and maintained by the National Centre for Biomedical Ontology (2012). Therefore, the ICF Ontology extends our Domain Ontology, which allows the class *dsml4ptm:ICFStandard* to be mapped with the concepts of the ICF Ontology. The language engineer creates four semantic mappings (object properties), which point to the four ICF main categories: *icf:BodyFunction*, *icf:ActivitiesAndParticipation*, *icf:EnvironmentalFactors*, and *icf:BodyStructures*. For this, SPARQL Rule 3a is instantiated (see Table 39).

Table 39. Instance of SPARQL Rule 3a

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
PREFIX dsml4ptm: <http://fhnw.ch/modellingEnvironment/DSML4PTM#>
PREFIX icf: <http://who.int/icf#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
INSERT DATA {
//creation of the new object properties for the mapping with domain ontology concepts (semantic mappings)
dsml4ptm:isMappedWithBodyFunction rdf:type rdf:Property .
dsml4ptm:isMappedWithBodyFunction rdfs:domain dsml4ptm:ICFStandard .
dsml4ptm:isMappedWithBodyFunction rdfs:range icf:BodyFunction .
dsml4ptm:isMappedWithBodyFunction rdfs:subPropertyOf lo:isMappedWithDOConcept .
dsml4ptm:isMappedWithActivityAndParticipation rdf:type rdf:Property.
dsml4ptm:isMappedWithActivityAndParticipation rdfs:domain dsml4ptm:ICFStandard .
dsml4ptm:isMappedWithActivityAndParticipation rdfs:range icf:ActivitiesAndParticipation .
dsml4ptm:isMappedWithActivityAndParticipation rdfs:subPropertyOf
lo:isMappedWithDOConcept .
dsml4ptm:isMappedWithEnvironmentalFactors rdf:type rdf:Property .
dsml4ptm:isMappedWithEnvironmentalFactors rdfs:domain dsml4ptm:ICFStandard .
dsml4ptm:isMappedWithEnvironmentalFactors rdfs:range icf:EnvironmentalFactors .
dsml4ptm:isMappedWithEnvironmentalFactors rdfs:subPropertyOf lo:
lo:isMappedWithDOConcept.
dsml4ptm:isMappedWithBodyStructures rdf:type rdf:Property .
dsml4ptm:isMappedWithBodyStructures rdfs:domain dsml4ptm:ICFStandard .
dsml4ptm:isMappedWithBodyStructures rdfs:range icf:BodyStructures.
dsml4ptm:isMappedWithBodyStructure rdfs:subPropertyOf lo:isMappedWithDOConcept .
}

```

Finally, the language engineer is requested to accommodate the additional requirement of assigning a performance level to each of the above-mentioned ICF categories. Therefore, the class *icf:Qualifier* of the ICF Ontology is specified by creating a new class *icf:Performance*. For this, the SPARQL Rule 4 is instantiated (see Table 40). This rule instance creates the new concept in the Domain Ontology as a sub-class of the existing class *icf:Qualifier*. In a later stage the *icf:Performance* can be further specified with levels such as no difficulty, mild difficulty, moderate difficulty, severe difficulty, complete difficulty, not specified and not applicable.

Table 40. Instance of SPARQL Rule 5

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX icf: <http://who.int/icf#>
INSERT DATA {
// creation of new domain ontology concept icf:Performance
icf:Performance rdf:type rdfs:Class .
icf:Performance rdf:label "ICF Qualifier Performance" .
icf:Performance rdfs:subClassOf icf:ICFQualifier .
}
```


5.3.4.3 Rules 6 and 7: Delete Modelling Constructs and Properties

The semantic rules 6 and 7 aim to support the deletion of modelling constructs, attributes and relations from an abstract syntax. An informal semantic rule is presented for deleting the modelling constructs, which also includes property deletion and one specifically for deleting properties. Then, the informal rules are transformed into SPARQL rules 6 and 7 respectively and finally they are instantiated.

5.3.4.3.1 Informal Semantic Rule 6 to Delete Modelling Constructs

The informal semantic rule for deleting a modelling construct deletes classes and their annotations, object and datatype properties from the Modelling Language Ontology. Also, the instances associated to the modelling constructs and the properties of the instance (i.e. label, graphical notations, object and datatype properties) are deleted from the Palette Ontology. By doing so, we ensure consistency of knowledge between the Modelling Language Ontology and the Palette Ontology, i.e. between the abstract syntax and the notation.

This rule supports Operator 3: Delete sub-class.

5.3.4.3.2 Informal Semantic Rules 7 to Delete Properties

In order to delete only the properties of a modelling construct, the informal semantic rule 7 is proposed. This rule allows deleting semantic mappings, bridging connectors and datatype properties of classes reflecting modelling constructs, which reside in the Modelling Language Ontology. In contrast to rule 6, rule 7 only affects the Modelling Language Ontology because the properties deleted by rule 7 only reflect aspects of the abstract syntax.

This rule supports Operator 6: Delete relation and Operator 9: Delete Attribute.

5.3.4.3.3 SPARQL Rules 6 to Delete Modelling Constructs

The informal semantic rule 6 is transformed into SPARQL Rule 6 and shown in Table 41. The DELETE WHERE statement is used to delete all the triples that have both the class *lo:ModellingConstructToDelete* and the related palette instance *po:PaletteElementInstanceToDelete* as subjects. This statement allows deleting not only the class and instance, but also their properties. Both *ModellingConstructToDelete* and *PaletteElementInstanceToDelete* will be replaced with concrete resources when SPARQL Rule 6 is instantiated.

Table 41. SPARQL Rule 6 – Delete modelling construct

```
PREFIX lo: < http://fhnw.ch/modellingEnvironment/LanguageOntology#>
PREFIX po: <http://fhnw.ch/modellingEnvironment/PaletteOntology#>
DELETE WHERE {
  lo:ModellingCosntructToDelete ?predicate ?object .
  po:PaletteElementInstanceToDelete ?predicate ?object .
}
```

5.3.4.3.4 SPARQL Rules 7 to Delete Properties

The informal semantic rule 7 is transformed into SPARQL Rule 7 and shown in Table 42. The DELETE WHERE statement is used to delete all the triples that have *PropertyToDelete* as their subject. *PropertyToDelete* are all object properties or datatype properties that are selected to be removed. The statement deletes the property and any values. The property *lo:PropertyToDelete* will be replaced with a concrete resource when instantiating SPARQL Rule 7.

Table 42. SPARQL Rule 7 – Delete property

```
PREFIX lo: <http://fhnw.ch/modellingEnvironment/LanguageOntology#>
DELETE WHERE {
  lo:PropertyToDelete ?predicate ?object .
}
```

An alternative to the above SPARQL Rule 7 is presented in Table 43. However, the one in Table 42 is preferred as it requires less lines of specification for the same result.

Table 43. Alternative to SPARQL Rule 7

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX lo: <http://fhnw.ch/modellingEnvironment/LanguageOntology#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
DELETE DATA
{
  lo:PropertyToDelete rdf:type rdf:Property .
  lo:PropertyToDelete rdfs:domain lo:SourceModellingConstruct .
  lo:PropertyToDelete rdfs:range lo:TargetModellingCosntruct .
}
```

5.3.4.3.5 Instantiation of SPARQL Rules 6 and 7

In order to instantiate SPARQL rules 6 and 7 to delete a resource, the same use case introduced in Sub-section 5.3.4.2.9 is considered. In particular, we build upon the results of the modelling language extension shown in Figure 117.

We assume that the language engineer deletes the class *dsml4ptm:ICFStandard*. For this, SPARQL Rule 6 is instantiated and shown in Table 44. In result, the class, all its properties, and the related instance *po:ICFStandard* (including the properties associated to the instance) are removed from the DSML.

Table 44. An instance of SPARQL Rule 6

```
PREFIX po: <http://fhnw.ch/modellingEnvironment/PaletteOntology#>
PREFIX dsml4ptm: <http://fhnw.ch/modellingEnvironment/DSML4PTM#>
DELETE WHERE {
  dsml4ptm:ICFStandard ?predicate ?object .
  po:ICFStandard ?predicate ?object .
}
```

Alternatively, if the language engineer had to delete the property *dsml4ptm:ICFStandardIsPartOfKoGuDataObject*, the SPARQL rule instance in Table 45 would be instantiated. That means, only the bridging connector between “*dsml4ptm:ICFStandard*” and “*dsml4ptm:KoguDataObject*” would be deleted.

Table 45. An instance of SPARQL Rule 7

```
PREFIX dsml4ptm: < http://fhnw.ch/modellingEnvironment/DSML4PTM#>
PREFIX po: <http://fhnw.ch/modellingEnvironment/PaletteOntology#>
DELETE WHERE {
  dsml4ptm:ICFStandardIsPartOfKoGuDataObject ?predicate ?object .
}
```

The results from the execution of the two rule instances can be found in Appendix D: Validation SPARQL Rules, folder D2.

5.3.4.4 Rules 8 to 11: Update Modelling Constructs and Properties

The semantic rules from 8 to 11 aim to update modelling constructs and properties. Updating a modelling construct may consist of changing annotation properties such as label, comment, and graphical notations. Updates may also apply to object and datatype properties, where not only labels are changed but also the range of both datatype and object properties. The name of a class or property refers to a label, which is specified with the property *rdfs:label*.

An update of the range of datatype properties includes changing value types to concrete values. Moreover, the language engineer may want to hide certain modelling constructs, which can be done by changing the value of the datatype property “*po:paletteConstructIsHiddenFromPalette*”.

An update of the range of object properties includes changing the semantic concepts for the given semantic mappings or changing the bridging concept for the given bridging connectors.

An update of a modelling construct and properties is performed by first deleting the resource and then by inserting a new one.

In the following, three informal semantic rules are first described: for updating a modelling construct, datatype properties and object properties, respectively. Then the informal rules are transformed into SPARQL and finally instantiated.

5.3.4.4.1 Informal Semantic Rule 8 to Update Modelling Constructs

The informal rule to update a modelling constructs consists of first deleting resources and then inserting new ones. The affected resources are the following:

- The annotation properties of the class (*ModellingConstructToUpdate*) such as label and comment from in the Modelling Language Ontology.
- The instance (*PaletteElementInstanceToUpdate*) in the Palette Ontology that is related to *ModellingConstructToUpdate*.
- Labels and values of the of the graphical notations for the palette and the model (*nameOfPaletteThumbnail* and *nameOfModelImage*), which also reside in the Palette Ontology.

This rule supports Operator 2: Update class.

5.3.4.4.2 Informal Semantic Rule 9 to Update Object Properties

This informal rule deletes and insert labels and ranges assigned to the object properties of a modelling construct. The rule only affects the Modelling Language Ontology.

This rule supports Operator 5: Update relation.

5.3.4.4.3 Informal Rule 10 and 11 to Update Datatype Properties

The informal rule 10 deletes and inserts labels, value types and concrete values assigned to the datatype properties of a modelling construct. This rule affects the Modelling Language Ontology alone.

The informal rule 11 deletes and inserts the Boolean value of the datatype property *po:paletteConstructIsHiddenFromPalette*. Updating the property value to *true* means hiding the graphical notation from the palette.

This rule supports Operator 10: Update attribute.

5.3.4.4.4 SPARQL Rule 8 to Update Modelling Constructs

The informal semantic rule 8 is transformed into SPARQL Rule 8, which is shown in Table 46. The upper part of Table 46 shows the DELETE DATA statement while the bottom of Table 46 shows the INSERT DATA statement. Both statements are executed in sequence and they both share the same prefixes. The modelling construct with its property values will be replaced with concrete resources when SPARQL Rule 8 is instantiated.

Table 46. SPARQL Rule 8 – Update modelling construct

```
PREFIX lo: <http://fhnw.ch/modellingEnvironment/LanguageOntology#>
PREFIX po: <http://fhnw.ch/modellingEnvironment/PaletteOntology#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
DELETE DATA{
lo:ModellingConstructToUpdate rdfs:label "name of class to delete" .
lo:ModellingConstructToUpdate rdfs:comment "description to delete" .
po:PaletteElementInstanceToUpdate rdfs:label "name of instance to delete" .
po:PaletteElementInstanceToUpdate po:paletteConstructHasPaletteThumbnail "name graphical
notation to delete" .
po:PaletteElementInstanceToUpdate po:paletteConstructHasModelImage "name graphical notation
to delete" .
}
INSERT DATA{
lo:ModellingConstructToUpdate rdfs:label "new name of class" .
lo:ModellingConstructToUpdate rdfs:comment "new description" .
po:PaletteElementInstanceToUpdate rdfs:label "new name of instance" .
po:PaletteElementInstanceToUpdate po:paletteConstructHasPaletteThumbnail "new name of the
selected graphical notation" .
po:PaletteElementInstanceToUpdate po:paletteConstructHasModelImage "new name of the
selected graphical notation" .
}
```

5.3.4.4.5 SPARQL Rule 9 to Update Object Properties

The informal semantic rule 9 is transformed into SPARQL Rule 9 and shown in Table 57. Like for SPARQL 8, the DELETE DATA statement is followed by the INSERT DATA statement. The object resource of the properties is either in the Modelling Language Ontology or in the Domain Ontology (see *lo:ConceptToReplace* and *do:ConceptToReplace* in Table 57). All object resources are first deleted and then new values are inserted. All the object resources are replaced with concrete resources when the SPARQL Rule 9 is instantiated.

Table 47. SPARQL Rule 9 – Update object property

```
PREFIX lo: <http://fhnw.ch/modellingEnvironment/LanguageOntology#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX do: <http://fhnw.ch/modellingEnvironment/DomainOntology#>
DELETE DATA{
lo:ObjectPropertyToUpdate rdfs:label "name to replace" .
lo:ObjectPropertyToUpdate rdfs:SubPropertyOf property for the SemanticMapping or
BridgingConnector .
// lo:ConceptToReplace or do:ConceptToReplace
lo:ObjectPropertyToUpdate rdfs:range lo:ConceptToReplace .
}
INSERT DATA{
lo:ObjectPropertyToUpdate rdfs:label "new name" .
lo:ObjectPropertyToUpdate rdfs:SubPropertyOf property for the SemanticMapping or
BridgingConnector .
// lo:ConceptToReplace or do:ConceptToReplace
lo:ObjectPropertyToUpdate rdfs:range lo:NewConcept .
}
```

5.3.4.4.6 SPARQL Rule 10 to Update Datatype Properties

The informal semantic rule 10 is transformed into SPARQL Rule 10 to update and datatype the property. The rule is described in Table 48. The DELETE DATA statement is followed by the INSERT DATA statement. The object resources to be replaced are *label* and *value type*. Thus, the object resources of the datatype properties are first deleted, and then new ones are inserted in the Modelling Language Ontology. All the object resources will be replaced with concrete resources when instantiating SPARQL Rule 10.

Table 48. SPARQL Rule 10 – Update datatype property

```

PREFIX lo: <http://fhnw.ch/modellingEnvironment/LanguageOntology#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
DELETE DATA{
lo:DataTypePropertyToUpdate rdfs:label "name to replace" .
// "value to replace"^^datatype to replace or datatype to replace
lo:DataTypePropertyToUpdate rdfs:range xsd:"value type to replace" .
}
INSERT DATA{
lo:DataTypePropertyToUpdate rdfs:label "new name" .
// "value to replace"^^datatype to replace or datatype to replace
lo:DataTypePropertyToUpdate rdfs:range xsd:"new value type" .
}

```

Table 49 contains the SPARQL Rule 11, which updates the Boolean value of *po:paletteConstructIsHiddenFromPalette* from *false* to *true*. This rule only affect the Palette Ontology.

Table 49. SPARQL Rule 11 – Update the datatype property to hide a modelling construct

```

PREFIX po: <http://fhnw.ch/modellingEnvironment/PaletteOntology#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
DELETE DATA{
po:PaletteElementInstanceToUpdate po:paletteConstructIsHiddenFromPalette "false"^^xsd:boolean .
}
INSERT DATA{
po:PaletteElementInstanceToUpdate po:paletteConstructIsHiddenFromPalette "true"^^xsd:boolean .
}

```

5.3.4.4.7 Instances of SPARQL Rules 8 to 11

The instantiation of SPARQL rules from 8 to 11 also refer to the use case introduced in Sub-section 5.3.4.2.9. Also in this case we build upon the results of the modelling language extension shown in Figure 117.

In the following, one instance of each of the above-introduced SPARQL rules is presented.

To instantiate SPARQL Rule 8, we assume that the language engineer changes the property values of the modelling construct *dsml4ptm:ICFStandard*. The rule instance supporting such changes are reported in Table 50. The first statement deletes the old property values whereas the second one inserts the new property values that are entered by the language engineer.

Table 50. An instance of SPARQL Rule 8 - update modelling construct

```
PREFIX po: < http://fhnw.ch/modellingEnvironment/PaletteOntology#>
PREFIX dsml4ptm: <http://fhnw.ch/modellingEnvironment/DSML4PTM#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
DELETE DATA {
  dsml4ptm:ICFStandard rdfs:label "ICF Standard Modelling Construct" .
  dsml4ptm:ICFStandard rdfs:comment "The ICF Standard describes..." .
  po:ICFStandard rdfs:label "ICF Standard" .
  po:ICFStandard po:paletteConstructHasPaletteThumbnail "ICFStandardForPalette.png" .
  po:ICFStandard po:paletteConstructHasModelImage "ICFStandardForModel" .
}
INSERT DATA {
  dsml4ptm:ICFStandard rdfs:label "ICF International Standard" .
  dsml4ptm:ICFStandard rdfs:comment "The ICF Standard describes..." .
  po:ICFStandard rdfs:label "ICF International Standard" .
  po:ICFStandard po:paletteConstructHasPaletteThumbnail "NewICFStandardForPalette.png" .
  po:ICFStandard po:paletteConstructHasModelImage "NewICFStandardForModel" .
}
```

To instantiate SPARQL Rule 9, we assume that the language engineer changes the *label* and *range* of the object property *dsml4ptm:isPartOf*. This object property has the *domain* *dsml4ptm:ICFStandard* and *range* *dsml4ptm:KoGuDataObject*. The rule instance that supports this change is shown in Table 50.

Table 51. An instance of SPARQL Rule 9 - update object properties

```

PREFIX dsml4ptm: <http://fhnw.ch/modellingEnvironment/DSML4PTM#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
DELETE DATA {
dsml4ptm:ICFStandardIsPartOfKoGuDataObject rdfs:label "ICF Standard is part of KoGu Data Object" .
dsml4ptm:ICFStandardIsPartOfKoGuDataObject rdfs:range dsml4ptm:KoGuDataObject .
dsml4ptm:ICFStandardIsPartOfKoGuDataObject rdfs:SubPropertyOf
lo:elementHasBridgingConcept .
}
PREFIX dsml4ptm: <http://fhnw.ch/modellingEnvironment/DSML4PTM#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX bpmn: <http://ikm-group.ch/archiMEO/BPMN#>
INSERT DATA {
dsml4ptm:ICFStandardIsPartOfKoGuDataObject rdfs:label "ICF Standard is part of Data Object" .
dsml4ptm:ICFStandardIsPartOfKoGuDataObject rdfs:range bpmn:DataObject .
dsml4ptm:ICFStandardIsPartOfKoGuDataObject rdfs:SubPropertyOf
lo:elementHasBridgingConcept .
}

```

To instantiate SPARQL Rule 10, we assume that the language engineer changes the *label* and *range* of the datatype property *dsml4ptm:hasTimeStamp*. This datatype property has the *domain dsml4ptm:ICFStandard*. The rule instantiation supporting the update is reported in Table 52.

Table 52. An instance of SPARQL Rule 10 – update datatype properties

```

PREFIX dsml4ptm: <http://fhnw.ch/modellingEnvironment/DSML4PTM#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX bpmn: <http://ikm-group.ch/archiMEO/BPMN#>
DELETE DATA {
  dsml4ptm:hasTimeStamp rdfs:label "ICF Standard has Timestamp" .
  dsml4ptm:hasTimeStamp rdfs:range xsd:dateTime .
}
INSERT DATA {
  dsml4ptm:hasTimeStamp rdfs:label "ICF Standard Document has Day of Creation" .
  dsml4ptm:hasTimeStamp rdfs:range xsd:date .
}

```

Finally, to instantiate SPARQL Rule 11, we assume that the language engineer hides the graphical notation *ICF Standard* from the palette. The rule instance that supports this update is reported in Table 53.

Table 53. An instance of SPARQL 11 - hide a modelling construct

```

PREFIX po: <http://fhnw.ch/modellingEnvironment/PaletteOntology#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
DELETE DATA{
  po:ICFStandard po:paletteConstructIsHiddenFromPalette "false"^^xsd:Boolean .
}
INSERT DATA{
  po:ICFStandard po:paletteConstructIsHiddenFromPalette "true"^^xsd:Boolean .
}

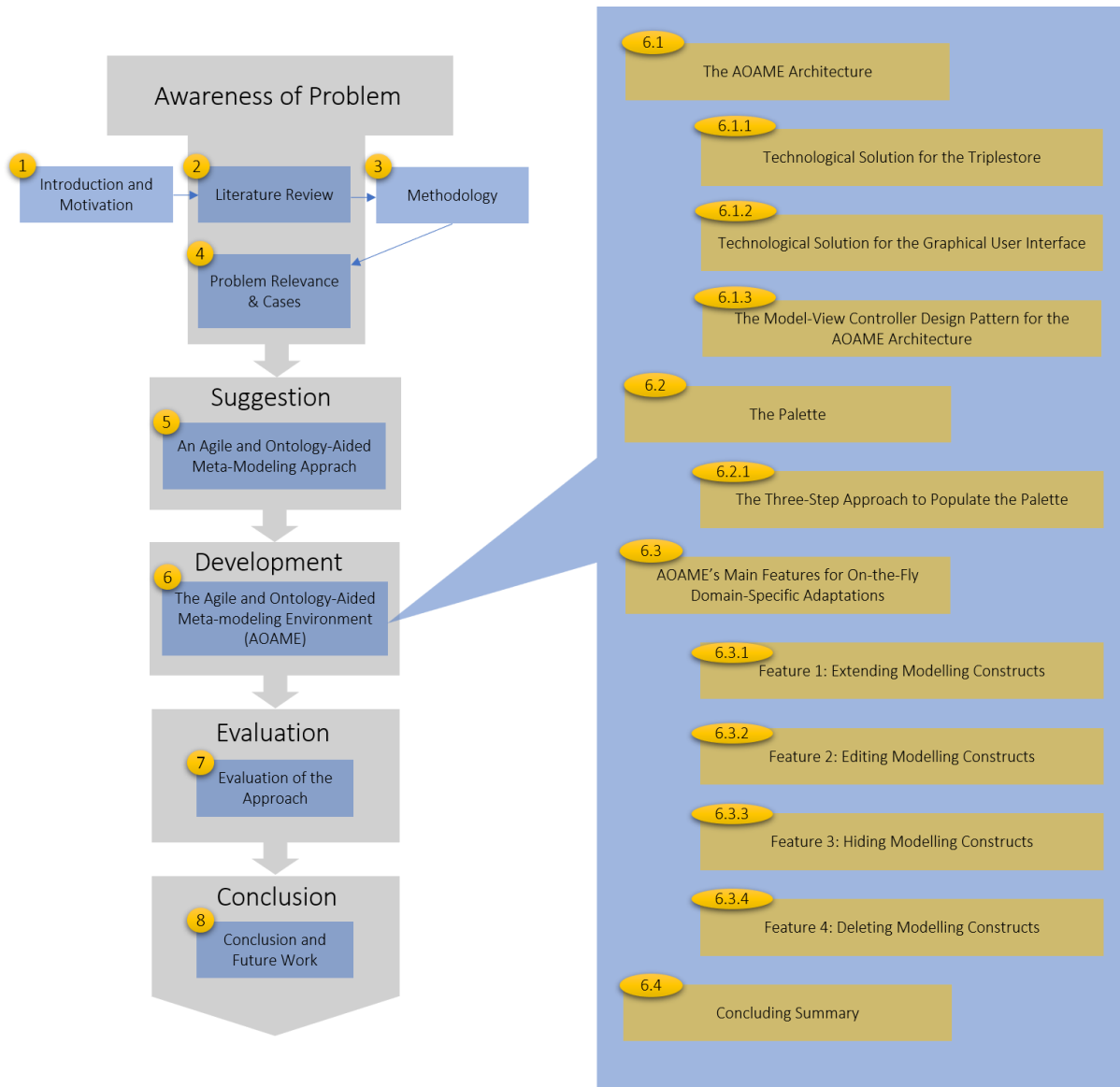
```

5.4 Concluding Summary

In this chapter a suggestion for an agile and ontology aided-meta-modelling approach is proposed. An agile meta-modelling was first proposed, which integrates the language engineering and modelling components into one. The integrated component allows the quick interleave of language engineering, modelling and evaluation activities. Therefore, the approach promotes the tight cooperation between the language engineer and the domain expert whilst simultaneously adaptations to a DSML can be accommodated on-the-fly. For the language adaptations, ten operators were derived which are applied in the integrated component. The approach has been further elaborated on by grounding modelling constructs with ontologies. Ontologies make the modelling language machine-interpretable. Thus, models built with such modelling languages have the benefit of being automated and with clear semantics. A new ontology-aided approach was conceived to support the agile meta-modelling. In particular, the ontology-aided approach includes an ontology-based meta-model architecture as well as eleven semantic rules. The semantic rules aim to preserve the alignment between the human-interpretable representation of a modelling language and its machine-interpretable representation. Rules were implemented in SPARQL and syntactically and semantically validated.

The agile and ontology-aided meta-modelling approach was published in (Laurenzi, Hinkelmann, Izzo, et al., 2018; Laurenzi, Hinkelmann, & van der Merwe, 2018) and answers the third research question.

6. THE AGILE AND ONTOLOGY-AIDED META-MODELLING ENVIRONMENT (AOAME)



This chapter describes the Agile and Ontology-Aided Meta-modelling Environment (AOAME). AOAME is the prototypical implementation of the agile and ontology-aided meta-modelling approach described in Chapter 5. The chapter addresses the fourth research question:

- (RQ4) How can the agile approach for domain-specific adaptations that preserves seamless consistency between the graphical and the machine-interpretable representation be automated?

The chapter is structured as follows: Section 6.1 provides an overview of the main components of the AOAME architecture and their interactions. The section continues by motivating and presenting the technology adopted for the front-end and back-end solutions. The section ends with the description of the Model-View-Controller design pattern, which was followed to conceive the architecture of the AOAME. Next, Section 6.2 introduces the Palette,

which displays the graphical notation of a modelling language. The palette provides access to various features that allow performing the domain-specific adaptations of modelling languages on-the-fly. Such features are described in Section 6.3 and ensure the continuous propagation of changes from the graphical representation of a modelling language to the ontology.

6.1 The AOAME Architecture

The AOAME architecture consists of three main components (see Figure 118):

- The triplestore contains the Modelling Language Ontology, the Palette Ontology and the Domain Ontology (Sub-section 5.3.2). The Domain Ontology can be extended with external ontologies or graphs. palette elements or connectors from the Palette Ontology have relationships with classes in the Modelling Language Ontology, which in turn have relationships with the classes in the Domain Ontology (see arrows among the three ontologies in Figure 118). The three ontology files can be found in Appendix E: Prototype Documentation, folder E4.
- The Graphical User Interface (GUI) is composed of the following two areas:
 - o The palette (left-hand side of the GUI) in which the graphical notations of an ontology-based meta-model are displayed, and it enables the domain-specific adaptations;
 - o The model editor (right-hand side of the GUI) in which models can be designed by selecting the graphical notations from the Palette. This research work focuses on the design or adaptation of modelling languages and DSMLs, therefore, the palette is further elaborated while the Model Editor is left to future work.

The source code for the GUI can be found in Appendix E: Prototype Documentation, folder E2.

- The Web service implements the logic to achieve the following two objectives:
 - o The propagation of the ontologies (including the graphical notations) from the triplestore to the GUI (see blue arrows in Figure 118). The propagation allows visualizing the ontology-based meta-model and notation in a human-interpretable representation in the GUI;
 - o The propagation of the domain-specific adaptations (or changes) from the GUI to the triplestore (see red arrows in Figure 118). The propagation is supported by the semantic rules introduced in Sub-section 5.3.3. The instances of the semantic rules are dynamically generated to transfer the language adaptations from the human- to the machine-interpretable representation.
- The Web service processes all the incoming and outgoing requests and manipulates data between the GUI and the triplestore. The code for the Web service can be found in Appendix E: Prototype Documentation, folder E3.

The following two sub-sections describe the technology solutions for the triplestore and the GUI.

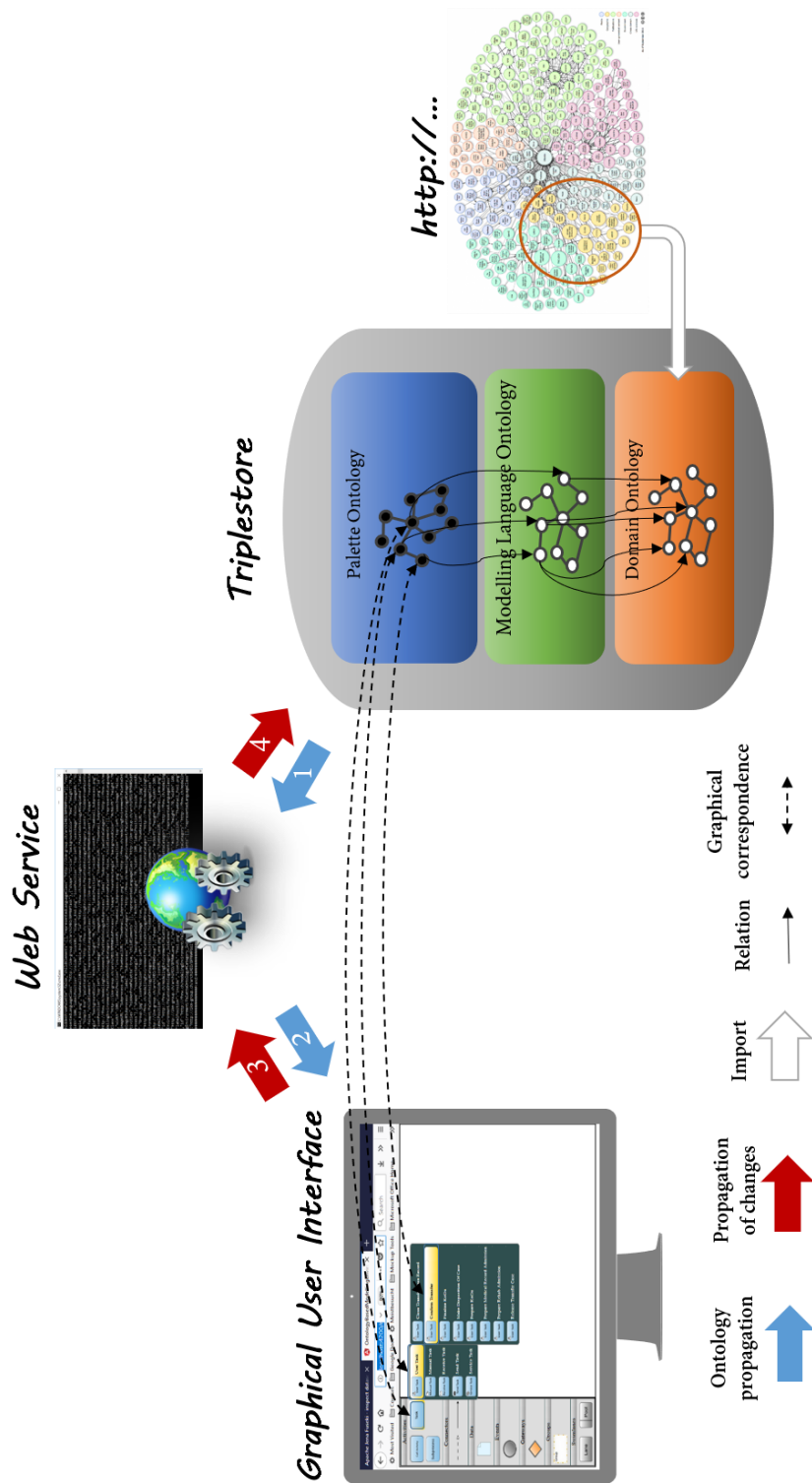


Figure 118. Components of the AOAME architecture

6.1.1 Technological Solution for the Triplestore

A triplestore (or triple store) is a purpose-built databases for the storage and retrieval of data as statements in the form of triples: *subject-predicate-object* (Allemang & Hendler, 2011). The supported ontology languages are commonly the Resource Description Framework (RDF), the RDF Schema RDF(S) and the Web Ontology Language (OWL). Triplestores also support the manipulation of ontologies, which is done through an inference engine. The latter is typically incorporated in a triplestore and offers reasoning services.

There exist various kinds of triplestores offered by different software vendors. Databases that store triples with a low expressive ontology language such as RDF have recently gained momentum, which led to create new sort of lightweight triplestores called graph databases or RDF databases. Nevertheless, triplestores are still commonly used and still more appropriate for dealing higher expressive ontologies such as RDF(S) and OWL. As already stated in Sub-section 5.3.2.2, the ontology language chosen for this work is RDF(S), therefore a triplestore is more suitable than an RDF database. The criteria for selecting a triplestore were the following:

- It should support the language ontology of RDF(S) to allow the instance of instance abstraction representation (a class being an instance simultaneously);
- It should support the execution of SPARQL rules introduced in Section 5.3.3. As a result, the chosen triplestore has to provide the SPARQL 1.1 protocols for update and for query.
- It should provide a user interface for server monitoring, dataset management (i.e. adding and removing ontologies) and query testing. The user interface with such services simplifies the test of the executed SPARQL rules from the user interface.
- It should be accessed and managed via Application Programming Interfaces (APIs).
- It should include an inference engine for RDFS entailments (W3C, 2004a). Although the inference support is out of scope in this research work, it is relevant for future research work. In particular, reasoning services are planned to be applied on ontology instances that reflect models built in the Model Editor area of AOAME.
- It should facilitate both the transformation between different data formats and the manipulation of data for the smooth propagation of knowledge between the GUI and the triplestore. The transformations are from JavaScript Object Notation (*JSON*) objects to the Turtle, the terse RDF Triple Language (“*.ttl*”) files and vice-versa.
- It should be open source given the scientific purpose and to avoid possible software vendor restrictions.

The chosen triplestore *Apache Jena Fuseki*⁵⁹ fulfils all the listed requirements.

⁵⁹ <https://jena.apache.org/documentation/fuseki2/index.html>

6.1.2 Technological Solution for the Graphical User Interface

The GUI is developed in Angular⁶⁰, which is an open-source framework embracing the Model-View-Controller (MVC) design pattern for web applications development (Leff & Rayfield, 2001). The MVC design pattern fosters the re-usability of developed components by adopting the separation of concerns principle by distinguishing among the following three interconnected parts:

- The model, which contains the objects that represent real-world entities;
- The view, which is what the user sees and interacts with. Requests are sent from the view and then the view shows the requested objects presented in the model;
- The controller, which implements the logic of the web application. First, it receives requests from the view. Then, the controller instructs the model to modify the containing objects or to prepare any information required by the view. In turns, the controller sends the requested information to the view to be shown to the user.

Angular has additional advantages in the context of the AOAME architecture:

- It ensures a rich and responsive experience to the user by providing data binding capability to HTML;
- It is maintained by Google and is an Opensource platform, which leads to an increasing supporting community of experts and users;
- It supports asynchronous calls, which allows multiple client requests simultaneously. Although it is not relevant for the current implementation of the artefact, it is relevant for the future evolution of AOAME.
- It uses Node.js in the backend, which simplifies the JQuery calls between the GUI and the Web service in AOAME.
- It is integrated with effective frameworks such as NativeScript⁶¹, which allows the creation of native apps in iOS and Android. That means, the code is written once and works in two different operating systems.
- Since it is a framework, it includes specific components by default, which facilitate the development. Therefore, the use of third-party libraries can be avoided, which otherwise could arise incompatibility issues. For example, Angular contains the “Modal” component (included in the “Material design components”), which supports the creation of the GUI in AOAME.

⁶⁰ <https://angular.io/>

⁶¹ <https://www.nativescript.org/>

6.1.3 The Model-View-Controller Design Pattern for the AOAME Architecture

The AOAME architecture follows the Model-View-Controller (MVC) design pattern (see the three big bubbles in light yellow, blue and green in Figure 119). Also the GUI follows the MVC design pattern (see the three black bubbles in the bottom left corner of Figure 119).

The *view* part of the GUI was implemented in Angular (see the *view* black bubble Figure 119) and is responsible to interact with the user by displaying (1) the graphical notations of a modelling language, (2) pop-up windows displaying knowledge of modelling constructs and (3) fields to enable the on-the-fly domain specific adaptations. Once the language engineer works in the GUI (e.g. extend a modelling element, creating a new relation, or edit an existing modelling relation), the *view* part sends the event to the *controller* part (see the *controller* black bubble in Figure 119), which takes it up and performs three actions:

- Manipulates the model part in Angular, which contains JSON objects that reflect the ontologies stored in the triplestore.
- Sends the request to the Web service (which is the controller of AOAME) through the use of dedicated services. The request is RESTful and performed via the HTTP protocol.
- Displays a new view in the GUI.

The two-way data binding between the *controller* part and *model* part enforces that each action received by the view is consistently reported in the model. The same applies between the *view* and the *model* of Angular, where each view displayed in the GUI is consistent with the model and its objects.

When the *controller* part of AOAME (see the light blue bubble in Figure 119) receives the requests containing the JSON objects, they are transformed into Java objects by the methods implemented in the Web Service. Next, the Java objects are manipulated in the Web service for the automatic generation of SPARQL rules (see Sub-section 5.3.3). The rules are then sent to the triplestore via HTTP methods, through which ontologies (i.e. the *model* part of AOAME, see the green bubble in Figure 119) get modified. The interaction between the Web service and the triplestore is supported by the Jena Application Programming Interfaces (APIs), which provide the needed interfaces for the communication. Further, the transformation of Java objects into an ontology format is implemented in the Web Service.

The logic in the Web service is also implemented to fetch the ontologies from the triplestore, which occurs as soon as the prototype starts. Therefore, the logic allows the initial propagation of the ontologies to the palette in the GUI. The model consisting of Java objects at the Web service is transformed into a model of JSON objects and is sent to the *view* part of AOAME (the big light yellow on the bottom left corner of Figure 119) over the http protocol. Then, dedicated services in the *view* part of AOAME, notify the *controller* part of Angular (see *controller* black bubble in Figure 119) such that it can generate a new view for the user as well as update the JSON objects in the *model* part of Angular.

The presented implemented architecture ensures the seamless alignment between the human- and the machine-interpretable representation of a modelling language, while domain-specific adaptations are performed on-the-fly on a modelling language.

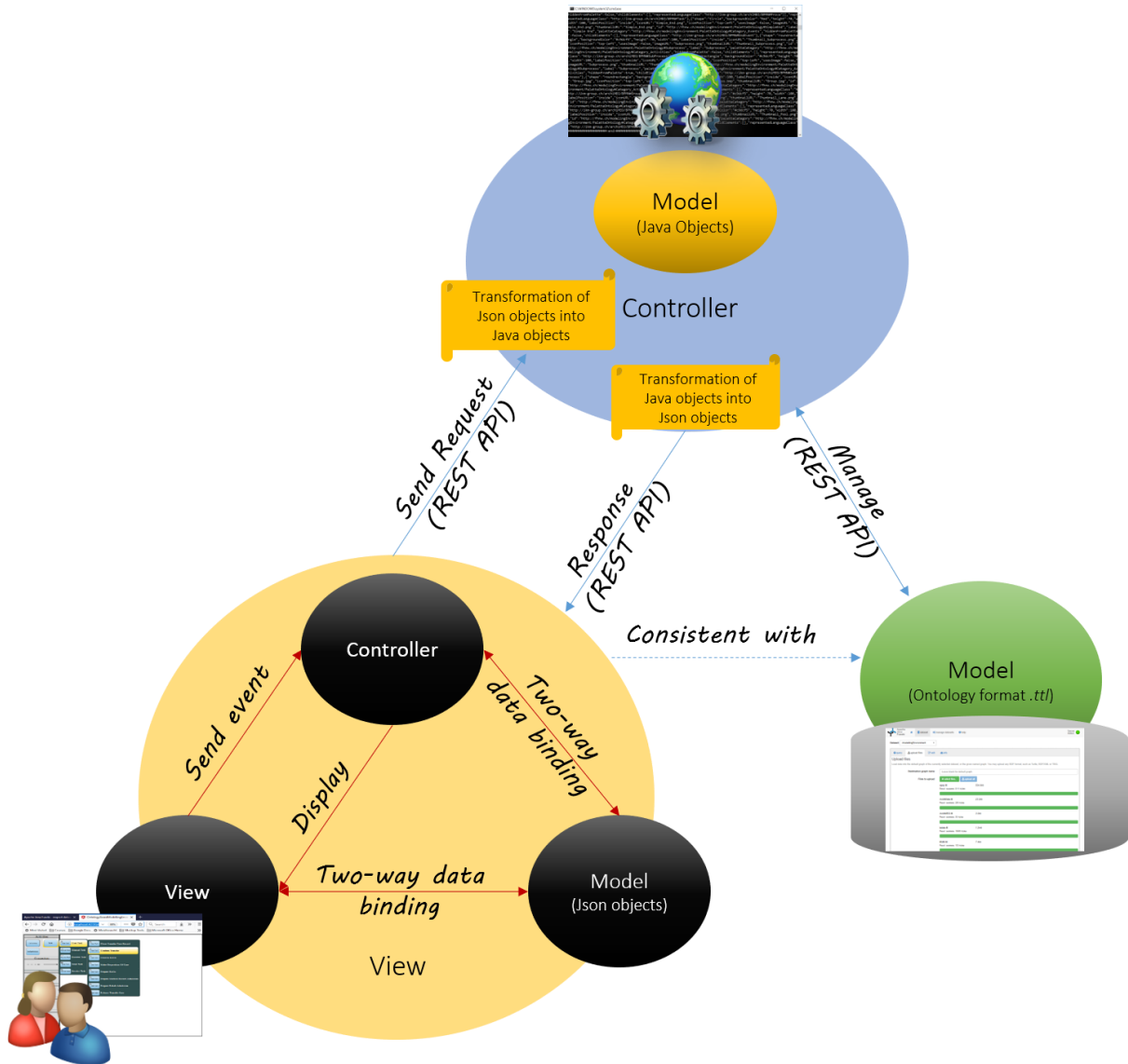


Figure 119. Model-View-Controller (MVC) design pattern in AOAME

6.2 The Palette

The palette has the purpose of displaying graphical notations of a modelling language that can be used for modelling. The logic for displaying the graphical notations is implemented in the Web Service. The visualisation of the graphical notations depends on the interaction between the Web service and both the Fuseki Triplestore and the GUI. Such interaction is made possible by Java methods implemented in the Web Service. In this section, demo excerpts are introduced to provide an understanding of the method components. Moreover, the correspondent names of endpoints are provided along with the text, which allows the quick identification of the correspondent Java method in the Web Service. The complete set of methods can be found in the Java class *ModellingEnvironment.java* in the package *ch.fhnw.modeller.webservice* (see Appendix E: Prototype Documentation, folder E3).

The following two sub-sections describe the mechanisms of the palette and its main implemented features. The latter allows the ontology-based meta-model to be adapted on-the-fly.

6.2.1 The Three-Step Approach to Populate the Palette

For the palette to be populated this research conceived the following sequential three-step approach:

- (1) Upload a set of ontologies;
- (2) Select a modelling language and a related modelling view;
- (3) Display graphical notations of the chosen modelling view.

The selection of both modelling language and modelling view is a useful feature that contributes to the agile approach. The feature allows the language engineer to quickly select the modelling languages to adapt and to switch among the different modelling views.

Figure 120 shows the three-step approach in the form of screenshots to underpin the following description:

1. Firstly, a set of ontologies should be uploaded on the Fuseki Triplestore.
2. Secondly, a modelling language can be selected from a dropdown list on the top of the palette. Once one modelling language is selected, the related modelling view(s) appear in the second dropdown list for selection.
3. The selection of the modelling view triggers the population of the palette with the graphical notations that belong to the chosen modelling view.

As an example, the rightmost screenshot of Figure 120 shows the graphical notations that belong to the *process modelling view* of the BPaaS extension of BPMN in Section 4.2. Each graphical notation is displayed in the category to which it belongs. The example shows seven categories grouping the graphical notations of the chosen modelling view: Activities, Connectors, Data, Events, Gateways, Groups and Swimlanes.

The required set of ontologies is further elaborated in Sub-section 6.2.1.1, while the logic for retrieving and displaying the graphical notations are described in Sub-section 6.2.1.2.

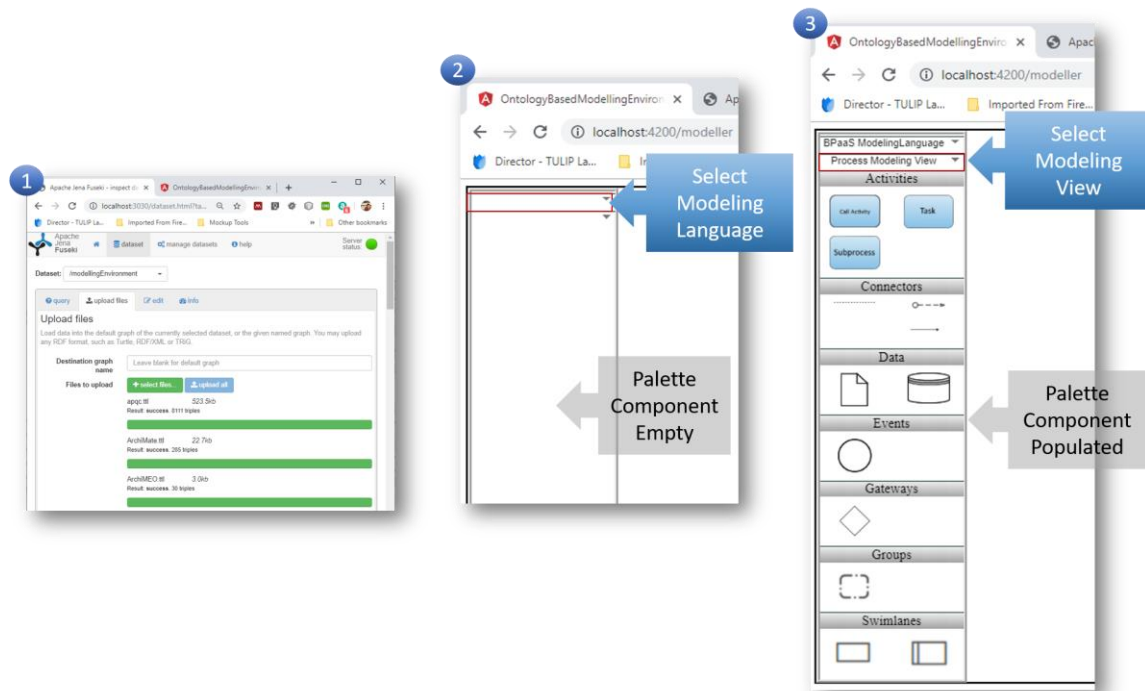


Figure 120. Three steps to populate the Palette

6.2.1.1 Required Set of Ontologies

The set of ontologies to be uploaded on the Fuseki Triplestore are the following:

- The Palette Ontology, the Modelling Language Ontology and the Domain Ontology are the three main ontologies.
- At least one ontology corresponding to the modelling language that it sought to be extended, e.g. BPMN ontology, CMMN ontology or ArchiMEO (Hinkelmann et al., 2013). If more than one modelling language is uploaded, the language engineer can integrate elements from the different languages or create bridging connectors between elements of the two languages.
- One or more domain ontologies to be used for the semantic mapping. The Domain Ontology enables the language engineer to create semantic mappings from the modelling elements to the domain ontology concepts. Additional domain ontologies that can extend the Domain Ontology and can be used for semantic mapping. This allows the reuse of already existing ontologies, e.g. biomedical ontology (National Center for Biomedical Ontology, 2012).

The set of ontologies is conceived to be prepared before it is uploaded. Additionally, all the graphical notations to be displayed in the Model Editor and the palette are stored in the Angular web application. The graphical notations can then be selected on-the-fly to be associated to new or existing modelling constructs, and subsequently they can be used in the Model Editor for the creation of models.

Future technological improvements will focus on the simplification of the uploading of new graphical notations. The idea is to allow a user to upload an image or picture directly on a cloud service to make it available as a new graphical notation to be chosen. The CSS implemented in the GUI already foresees the automatic re-size of the images. Therefore, additional work on resizing an image to be displayed in the palette is avoided.

6.2.1.2 Logic for Retrieving and Displaying Graphical Notations in the Palette

The logic of retrieving and displaying the graphical notations in the palette is distributed between the Web service and the Angular Web Application, which implements the GUI. According to the selected modelling language and view, a set of categories is identified, which is then matched with the graphical notations to display. While the knowledge retrieval is performed by queries that are dynamically generated in the Web Service, the mapping is implemented in the Angular Web Application⁶². In the following, the implemented logic is described in detail.

The Web service contains four endpoints: (1) *getModellingLanguage*, (2) *getModellingView*, (3) *getPaletteCategories* and (4) *getPaletteElement*. Each endpoint implements a method that allows the automatic generation of SPARQL SELECT⁶³ queries.

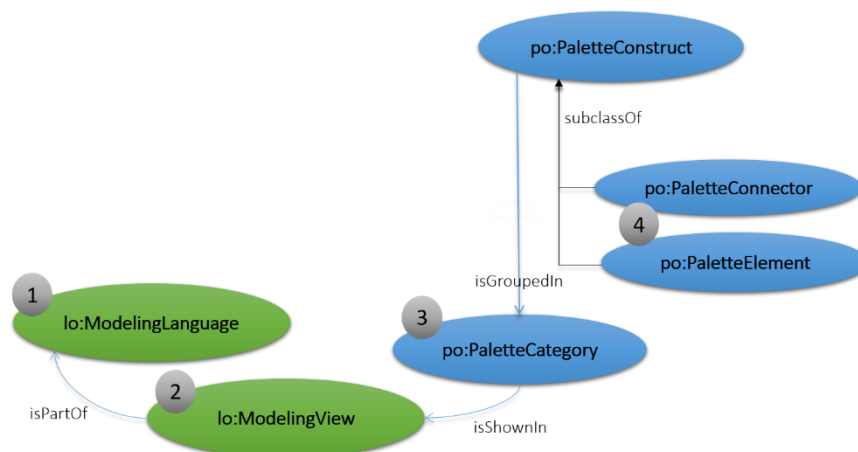


Figure 121. Correspondence between the four endpoints and the ontology structure

The correspondence between the generated semantic queries and the affected classes in the ontology is graphically represented in Figure 121.

1. From the first endpoint *getModellingLanguage* a semantic query is generated to retrieve all instances of the class *lo:ModellingLanguages*, which contains the modelling languages that are available in the ontology.
2. Subsequently, from the endpoint *getModellingView* the generated semantic query retrieves those instances of the class *lo:ModellingView* that are part of the selected modelling language. Figure 122 shows two snippets of the Java methods implemented in the two endpoints. Note that in the second snippet (*getModellingView*), the query includes the parameter *langId*, which intends to contain the *id* of the selected modelling language.

⁶² The source code for the matchmaking can be found under the path folder *src\app\palette-area\palette-area.component.html* of the Angular Web Application from row 15 to row 29.

⁶³ The source code for the endpoints can be found in the Java class “*ModellingEnvironment.java*” of the package “*ch.fhnw.modeller.webservice*” of the Web service from row 49 to row 443.

(1) @getModelingLanguage

```
private ArrayList<ModelingLanguage> queryAllModelingLanguages() throws NoResultsException {
    ParameterizedSparqlString queryStr = new ParameterizedSparqlString();
    ArrayList<ModelingLanguage> result = new ArrayList<ModelingLanguage>();
    queryStr.append("SELECT ?element ?label ?hasModelingView ?viewIsPartOfModelingLanguage ?type WHERE {");
    queryStr.append("?element rdf:type ?type . FILTER(?type IN (lo:ModelingLanguage)) .");
    queryStr.append("?element rdfs:label ?label . ");
    queryStr.append("}");
}
```

(2) @getModelingView

```
private ArrayList<ModelingView> queryAllModelingViews(String langId) throws NoResultsException {
    ParameterizedSparqlString queryStr = new ParameterizedSparqlString();
    ArrayList<ModelingView> result = new ArrayList<ModelingView>();
    queryStr.append("SELECT ?element ?label ?viewIsPartOfModelingLanguage ?type WHERE {");
    queryStr.append("?element rdfs:label ?label .");
    queryStr.append("?element lo:viewIsPartOfModelingLanguage "+ langId + " .");
    queryStr.append("}");
}
```

Figure 122. Snippets of the Java methods that generate SPARQL SELECTS for retrieving (1) modelling languages and (2) modelling view(s)

- Next, the third endpoint *getPaletteCategories* generates a query that retrieves those instances of the class *po:PaletteCategory* that are bounded with the chosen modelling view. The upper part of Figure 123 shows a snippet of the method that generates the query. The parameter “*viewId*” represents the *id* of the chosen modelling view. The retrieved categories are then used in the Angular Web Application for the mapping with the graphical notations to display. The latter are retrieved in the fourth endpoint *getPaletteElement*.
- Finally, the *getPaletteElement* endpoint implements the method that generates the semantic query to retrieve all instances of the two classes *po:PaletteConnector* and *po:PaletteElement* (in the Palette Ontology). The instances contain all the required knowledge for displaying the graphical notations. An excerpt of the method is shown in the lower part of Figure 123. Note that the method does not contain any *id* as the query is not bounded with the retrieved category. In fact, the number of categories and categories themselves may change each time a different modelling view is selected. Incorporating this logic in the query is not ideal as the query gets unnecessarily complicated. Instead, the logic of the mapping is applied in the front-end web application, where only the palette elements and connectors that match with the previously retrieved categories are displayed.

(3) @getPaletteCategory

```
private ArrayList<PaletteCategory> queryAllPaletteCategories(String viewId) throws NoResultsException {
    ParameterizedSparqlString queryStr = new ParameterizedSparqlString();
    ArrayList<PaletteCategory> result = new ArrayList<PaletteCategory>();
    queryStr.append("SELECT ?category ?label ?orderNumber ?hidden WHERE {");
    queryStr.append("?category rdf:type* po:PaletteCategory .");
    queryStr.append("?category rdfs:label ?label .");
    queryStr.append("?category po:paletteCategoryIsShownInModelingView " + viewId + " .");
    queryStr.append("OPTIONAL {?category po:paletteCategoryOrderNumber ?orderNumber .}");
    queryStr.append("OPTIONAL {?category po:hiddenFromPalette ?hidden .}");
    queryStr.append("}");
    queryStr.append("ORDER BY ?orderNumber");
}
```

(4) @getPaletteElement

```
private ArrayList<PaletteElement> queryAllPaletteElements() throws NoResultsException {
    ParameterizedSparqlString queryStr = new ParameterizedSparqlString();
    ArrayList<PaletteElement> result = new ArrayList<PaletteElement>();
    queryStr.append("SELECT ?element ?label ?representedClass ?hidden ?category ?categoryLabel ?parent ?backgr");
    queryStr.append("?element rdf:type ?type . FILTER(?type IN (po:PaletteElement, po:PaletteConnector)) .");
    queryStr.append("?element rdfs:label ?label .");
    queryStr.append("?element po:paletteConstructIsRelatedToModelingLanguageConstruct ?representedClass .");
    queryStr.append("?element po:paletteConstructIsHiddenFromPalette ?hidden .");
    queryStr.append("?element po:paletteConstructIsGroupedInPaletteCategory ?category .");
    queryStr.append("?category rdfs:label ?categoryLabel .");
    ...
}
```

Figure 123. Snippets of the Java methods that generate SPARQL SELECTS for retrieving (3) categories and (4) graphical notations

In order to provide a better understanding of what a dynamically generated query looks like and what results can produce, the following sub-section describes a query generated by the fourth endpoint *getPaletteElement*. The complete list of queries samples covering each of the introduced endpoints can be found in Appendix E: Prototype Documentation, folder E1.

6.2.1.2.1 An Example of a Dynamically Generated SPARQL SELECT

The upper part of Figure 124 shows an excerpt of the query, which is generated from the fourth endpoint *getPaletteElement*. The query was slightly adapted to show the result given a specific category. Therefore, the mapping between the identified categories and the palette elements is incorporated in the query in this example. In particular, the change affects the triple shown in row 13 (see red arrow) of Figure 124, where the generic object variable *?category* is replaced by a concrete instance of the class *po:PaletteCategory*: *po:Category_Activities4BPMNProcessModellingView*.

The generic object variable *category* serves to group the concepts about activities, which belong to the process modelling view. In the Palette Ontology, the category is shared by two modelling views: *lo:BPMNProcessModellingView* and *lo:BPaaSProcessModellingView*, which are the modelling views of BPMN and BPaaS, respectively. Note that the two views are kept separate as they diverge in some concepts. The elements and properties resulting from the execution of the query (see bottom of Figure 124) are used to display the graphical notations in the palette. Consequently, the retrieved information is transformed into JSON objects and streamed to the Angular web application. The fourth row of Figure 124 shows the properties of the User Task, which are the following:

- The task label (*User Task*);
- The belonging class (*po:PaletteElement*);
- The related modelling construct in the abstract syntax (*bpmn:User Task*);

- The value of the datatype property *isHiddenFromPalette* (“false”^{^^xsd:Boolean}, which makes the graphical notation visible);
- The belonging category (*po:Category_Activity*, to group the elements under the category Activity in the palette);
- The parent of the graphical notation hierarchy (*po:Task*). It is optional as root concepts do not have any parents;
- The image to be displayed in the Model Editor (*User_Task.png*);
- The thumbnail to be displayed in the palette (*Thumbnail_User_Task.png*);
- The default height of the image for the Model Editor (“70”^{^^xsd:integer}, with pixels as a unit of measure);
- The default width of the image for the Model Editor (“100”^{^^xsd:integer}, with pixels as a unit of measure).

```

8 SELECT DISTINCT ?element ?label ?type ?relatedClass ?categoryLab ?hidden ?parent ?image ?thumbnail ?height ?width
9 * WHERE {?element rdf:type ?type . FILTER(?type IN (po:PaletteElement, po:PaletteConnector)) .
10 ?element rdfs:label ?label .
11 ?element po:paletteConstructIsRelatedToModelingLanguageConstruct ?relatedClass .
12 ?element po:paletteConstructIsHiddenFromPalette ?hidden .
13 ?element po:paletteConstructIsGroupedInPaletteCategory po:Category_Activities4BPMNProcessModelingView .
14 ?element po:paletteConstructIsGroupedInPaletteCategory ?category .
15 ?category rdfs:label ?categoryLab .
16 * OPTIONAL{ ?element po:paletteConstructHasHeight ?height }.
17 * OPTIONAL{ ?element po:paletteConstructHasModelImage ?image }.
18 * OPTIONAL{ ?element po:paletteConstructHasPaletteThumbnail ?thumbnail }.
19 * OPTIONAL{ ?element po:paletteConstructHasWidth ?width }.
20 * OPTIONAL{ ?relatedClass rdfs:comment ?comment }.
21 * OPTIONAL{ ?element po:paletteConstructHasParentPaletteConstruct ?parent }.
22

```

Activity Category for Process Modeling View

element	label	type	relatedClass	categoryLab	hidden	parent	image	thumbnail	height	width
1 po:Activity	"Activity"	po:PaletteElement	<http://ikm-group.ch/archiMEO/BPMN#Activity>	"Activities"	"false"^^xsd:boolean		"Call_Activity.png"	"Thumbnail_Call_Activity.png"	"70"^^xsd:integer	"100"^^xsd:integer
2 po:BusinessRuleTask	"Business Rule Task"	po:PaletteElement	<http://ikm-group.ch/archiMEO/BPMN#BusinessRuleTask>	"Activities"	"false"^^xsd:boolean	po:Task	"Business_Rule_Task.png"	"Thumbnail_Business_Rule_Task.png"	"70"^^xsd:integer	"100"^^xsd:integer
3 po:Task	"Task"	po:PaletteElement	<http://ikm-group.ch/archiMEO/BPMN#Task>	"Activities"	"false"^^xsd:boolean		"Task.png"	"Thumbnail_Task.png"	"70"^^xsd:integer	"100"^^xsd:integer
4 po:UserTask	"User Task"	po:PaletteElement	<http://ikm-group.ch/archiMEO/BPMN#UserTask>	"Activities"	"true"^^xsd:boolean	po:Task	"User_Task.png"	"Thumbnail_User_Task.png"	"70"^^xsd:integer	"100"^^xsd:integer
5 po:ManualTask	"Manual Task"	po:PaletteElement	<http://ikm-group.ch/archiMEO/BPMN#ManualTask>	"Activities"	"false"^^xsd:boolean	po:Task	"Manual_Task.png"	"Thumbnail_Manual_Task.png"	"70"^^xsd:integer	"100"^^xsd:integer

Figure 124. Execution of a concrete query generated from the fourth endpoint “getPaletteElement”

6.3 AOAME's Main Features for On-the-Fly Domain-Specific Adaptations

There are four main features implemented in the palette of AOAME that enable the on-the-fly domain-specific adaptations of ontology-based meta-models:

- *Extending a Modelling Construct:* this feature allows for the extension of both the modelling constructs and properties;
- *Editing a Modelling Construct:* this feature allows for the editing of both modelling constructs and properties and for the removal of properties;
- *Hiding a Modelling Construct:* this feature allows hiding modelling constructs from the palette;
- *Deleting a Modelling Construct:* this feature allows for the removal of modelling constructs.

The functionalities are listed in Figure 125. The four main functionalities are displayed once the user right-clicks on one of the graphical notations. The palette is implemented to show the hierarchy of modelling constructs. If an element contains one or more sub-elements, they are shown in a new tier. The new tier is displayed as soon as the cursor is moved on one graphical notation. If a modelling construct is extended with a new one, the latter will appear in the $n + 1$ tier, where n is the number of tiers in which the construct being extended is contained.

Figure 125, for example, shows sub-elements of User Task, which in turn is the sub-element of Task. The knowledge about the hierarchy is captured in the Palette Ontology by the *isParentOf* object property placed between instances of the class Palette Construct. For example, in Figure 125, User Task is displayed in the second tier of the hierarchy because in the Palette Construct the instance *po:Task* is the parent of *po:UserTask*.

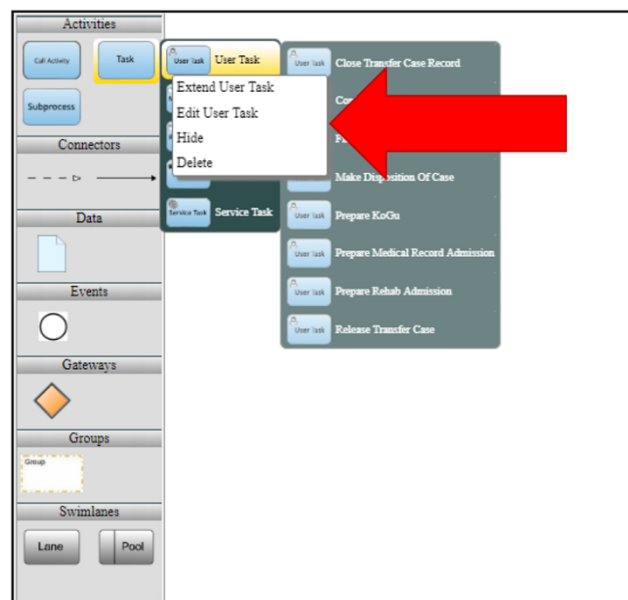


Figure 125. Main functionalities in the palette of AOAME

During the implementation of phase, it has been identified the convenience of distinguishing between the object properties and datatype properties also in the ontology terminology. Such distinction allowed the creation of methods for the dynamic generation of

SPARQL rules dealing with either object properties or datatype properties. Therefore, although the RDF(S) is adopted as an ontology language, properties are specified with the OWL terminology: *owl:ObjectProperty* and *owl:DatatypeProperty*. Such specifications do not affect the knowledge assumption of the adopted ontology language RDF(S), thus the created models would still benefit from the *Closed World Assumption* (CWA) and the *Unique Names Assumption* (UNA) on models (see Sub-section 5.3.2.2). Also, since both object properties and datatype properties are sub-properties of *rdf:Property*, the RDF(S) entailment patterns regarding properties remain valid, i.e. *rdfs5*, *rdfs6* and *rdfs7* (W3C, 2014b).

6.3.1 Feature 1: Extending Modelling Constructs

Figure 126 depicts the feature for extending a modelling construct. The view consists of a pop-up window, which appears after the selection of the extension feature (see red arrow) from the context menu, e.g. Extend User Task. The pop-up shows fields for and from the ontology-based meta-model. There are three types of fields:

- Non-interactive fields for an informative purpose (see grey boxes in Figure 126),
- Interactive fields that allow entering knowledge (see blue boxes in Figure 126),
- Interactive fields that lead to a different view (see green box in Figure 126).

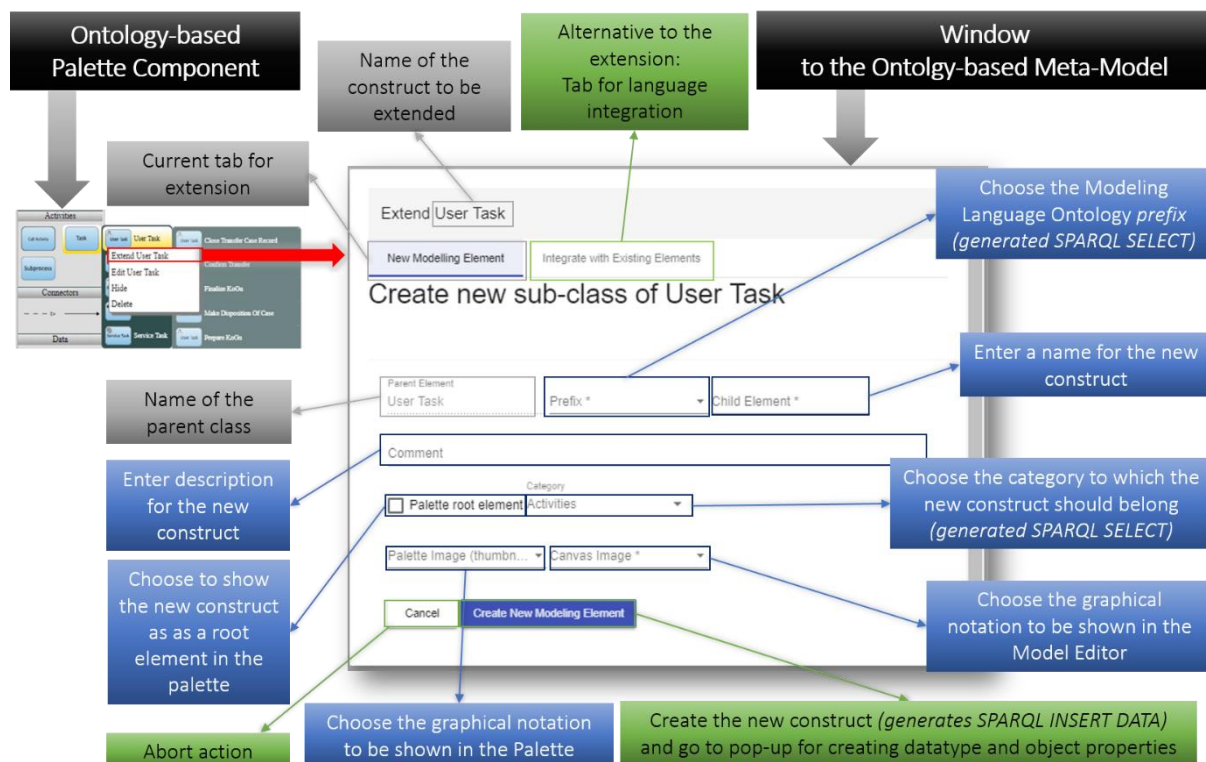


Figure 126. Main view for extending a modelling construct: Creation of the new class and its annotation properties

In the following, the field types in Figure 126 are detailed.

- The non-interactive and informative fields are the following:
 - The field to display the name of the construct to be extended,
 - The field to display the current tab for extension,
 - The field to display the name of the parent class. This field contains the same value as the field name of the construct to be extended.
- The interactive fields that allow entering knowledge are the following:
 - The *prefix* field is a mandatory field and consists of a dropdown list containing the namespaces of all language ontologies that are imported in the Modelling Language Ontology. This field generates a SPARQL SELECT of all the existing prefixes in the Modelling Language Ontology. For instance, in the ontology concept *bpmn:UserTask*, the prefix is *bpmn* and identifies the Business Process

Management and Notation Ontology, which is imported in the Modelling Language Ontology.

- The child element (or sub-element) field consists of a text box and allows entering the name of the new modelling construct. The content of this field is stored as the *rdfs:label* of the ontology concept. Filling this field is mandatory.
 - The comment field consists of a text box and allows entering the description for the new modelling construct. The content of this field is stored as the *rdfs:comment* of the ontology concept. Filling this field is mandatory.
 - The field for the palette root element consists of a check box to determine whether the modelling construct should appear in the class hierarchy (i.e. unticked check box) or in the palette (i.e. ticked check box). The check box is unticked by default.
 - The category field consists of a dropdown list and allows selecting the palette category for the new modelling construct. This field has a dependency on the palette root element, where the palette category is only considered if the check box is ticked as the new construct would need to be shown in the palette. This field generates a SPARQL SELECT of all the existing categories conceptualised for the modelling view of the current modelling construct. The implemented Java method can be found in the endpoint */getPaletteCategories* of the Web Service.
 - The graphical notation field for the palette consists of a dropdown list and allows selecting the graphical notation for the new construct to be shown by the palette. The selection of the graphical notation is mandatory.
 - The graphical notation field for the Model Editor consists of a dropdown list and allows selecting the graphical notation for the new construct to be shown in the Model Editor. The selection of the graphical notation is mandatory.
- (c) Among the interactive fields that lead to a different view, the following can be found:
- The integration tab, where the selected modelling construct can be extended with an existing Modelling Language Ontology concept. That is, the two modelling constructs are integrated with the *subClassOf* relation. The field consists of a tab, which, if clicked, leads to the view depicted in Figure 133.
 - The cancel button allows aborting the extension a modelling construct. By clicking to this button, the user is redirected to the main view of the GUI containing both the palette and the Model Editor.
 - The create a new modelling construct button allows creating a new construct as an extension of the selected one. The new construct is stored in the triplestore with the above-entered properties as well as some implicit ones, e.g. the modelling language and modelling view of belonging. The latter is not explicitly selected by the user but derived by the chosen modelling language (or modelling view), which occur right after launching the prototype. All the considered properties are listed in the SPARQL Rule 2 (INSERT DATA), which is described in Sub-section 5.3.3. The button leads to generate an instance of **SPARQL Rule 2**. When the palette is populated, the Web service implements Java method for the automatic generation of this SPARQL rule. The method is implemented in the endpoint depicted in Figure 127, where the complete code can be retrieved under the endpoint */createPaletteElement* in the Java class *ModellingEnvironment.java* of the Web Service. The two parts of the snippet in Figure 127 correspond to the parts that allow

entering knowledge in (1) the Modelling Language Ontology and (2) the Palette Ontology, respectively.

```

@POST
@Path("/createPaletteElement")
public Response insertPaletteElement(String json) {

    Gson gson = new Gson();
    PaletteElement pElement = gson.fromJson(json, PaletteElement.class);
    ParameterizedSparqlString querStr = new ParameterizedSparqlString();

    //Insert sub-class and annotation properties in the Modeling Language Ontology
    querStr.append("INSERT DATA {");
    querStr.append(pElement.getLanguagePrefix() + pElement.getUuid() + " rdf:type rdfs:Class .");
    querStr.append(pElement.getLanguagePrefix() + pElement.getUuid() + " rdfs:subClassOf <"+ pElement.getParentLanguageClass() + "> .");
    querStr.append(pElement.getLanguagePrefix() + pElement.getUuid() + " rdfs:label \"" + pElement.getLabel() + "\" .");
    //comment is optional
    if(pElement.getComment()!=null && !"".equals(pElement.getComment()))
    querStr.append(pElement.getLanguagePrefix() + pElement.getUuid() + " rdfs:comment \"" + pElement.getComment() + "\" .");
    //Insert instances and annotation properties in the Palette Ontology
    querStr.append("po:" + pElement.getUuid() + " rdf:type " + "http://fhnw.ch/modelingEnvironment/PaletteOntology#PaletteElement" + " .");
    querStr.append("rdfs:label \"" + pElement.getLabel() + "\" .");
    querStr.append("po:isRelatedToModelingConstruct " + pElement.getRepresentedLanguageClass() + " .");
    querStr.append("po:hasParentPaletteConstruct <" + pElement.getParentElement() + "> .");
    querStr.append("po:isGroupedInPaletteCategory <" + pElement.getPaletteCategory() + "> .");
    querStr.append("po:belongsToModelingLanguage " + pElement.getModelingLanguage() + " .");
    querStr.append("po:belongsToModelingView " + pElement.getModelingView() + " .");
    querStr.append("po:hasPaletteThumbnail \"" + pElement.getThumbnailURL() + "\" .");
    querStr.append("po:hasModelImage \"" + pElement.getImageURL() + "\" .");
    querStr.append("po:hasWidthForModel " + pElement.getWidth() + " .");
    querStr.append("po:hasHeightForModel " + pElement.getHeight() + " .");
    querStr.append("po:isHiddenFromPalette " + pElement.getHiddenFromPalette() + " .");
    querStr.append("}");
    querStr.append("}");
}

```

Figure 127. Excerpt of the method implemented in the Web service to generate the SPARQL INSERT DATA

The button “Create New Modelling Construct” simultaneously leads to a follow-up view. The new view contains both datatype and object properties to be entered (see Figure 128) and is elaborated below.

6.3.1.1 View to Deal with Datatype and Object Properties

After clicking the button “Create New Modelling Construct”, a view appears that deals with the datatype property and the two types of object properties: bridging connector and semantic mapping. For each of these property types there is a tab with the correspondent name. The first tab that is shown in the view (see left-hand side of Figure 129) deals the datatype property. The second- and third-tabs deal with the bridging connectors and the semantic mapping, respectively. Each tab contains two buttons:

- One button for inserting a new property and leads to a new view:
 - o insert a new datatype property leads to the view in Figure 129,
 - o insert a new bridging connector leads to the view in Figure 130,
 - o insert a new semantic mapping leads to the view in Figure 131.
- One button for aborting the action, which leads to the main view of AOAME (see Figure 125). That is, the newly created modelling construct will not contain its own datatype and object properties, but will contain:
 - o annotation properties entered in the view depicted in Figure 126;
 - o datatype and object properties inherited from the parent class.

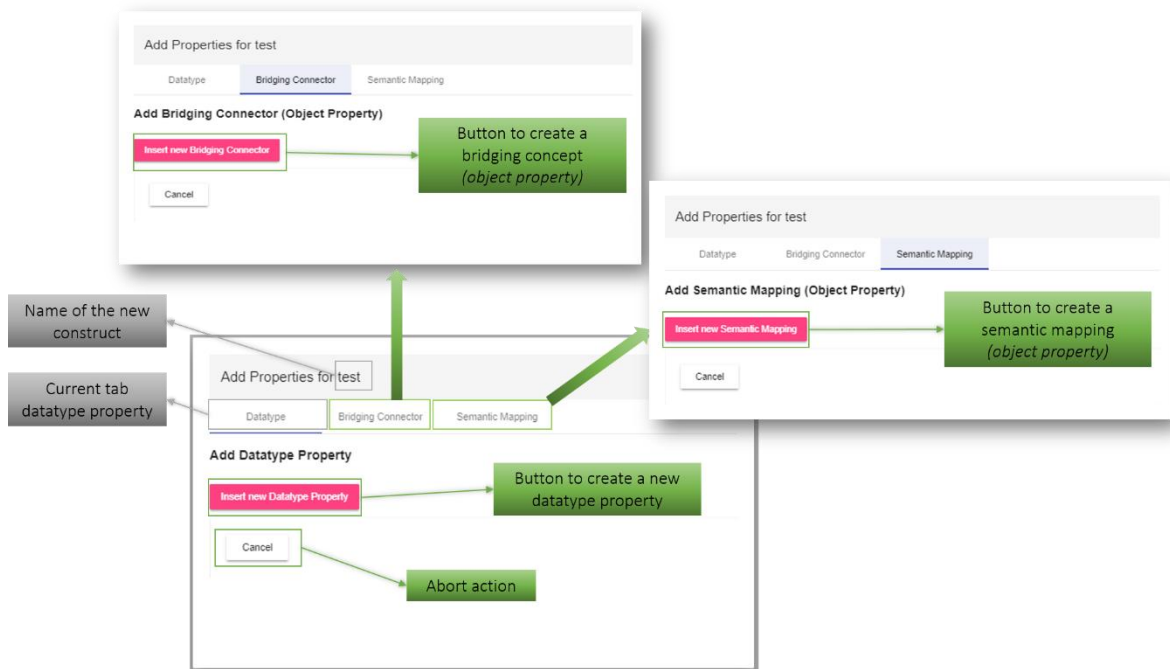


Figure 128. Adding datatype properties and object properties (i.e. bridging connectors and semantic mappings)

6.3.1.2 Creating New Datatype Properties

The right-hand side of Figure 129 shows the view for adding datatype properties. It appears after clicking to the button “Insert new Datatype Property”. The view foresees the following four fields:

- The label field allows entering the name for the datatype property.
- The value type field allows selecting a value type. The value type refers to the RDF-compatible XSD types such as Boolean, Date, DateTime, Decimal, Integer, String.
- The creating datatype property button leads to generate an instance of SPARQL Rule 4. The java method can be retrieved under the endpoint `/createDatatypeProperty` in the Java class `ModellingEnvironment.java` of the Web Service.

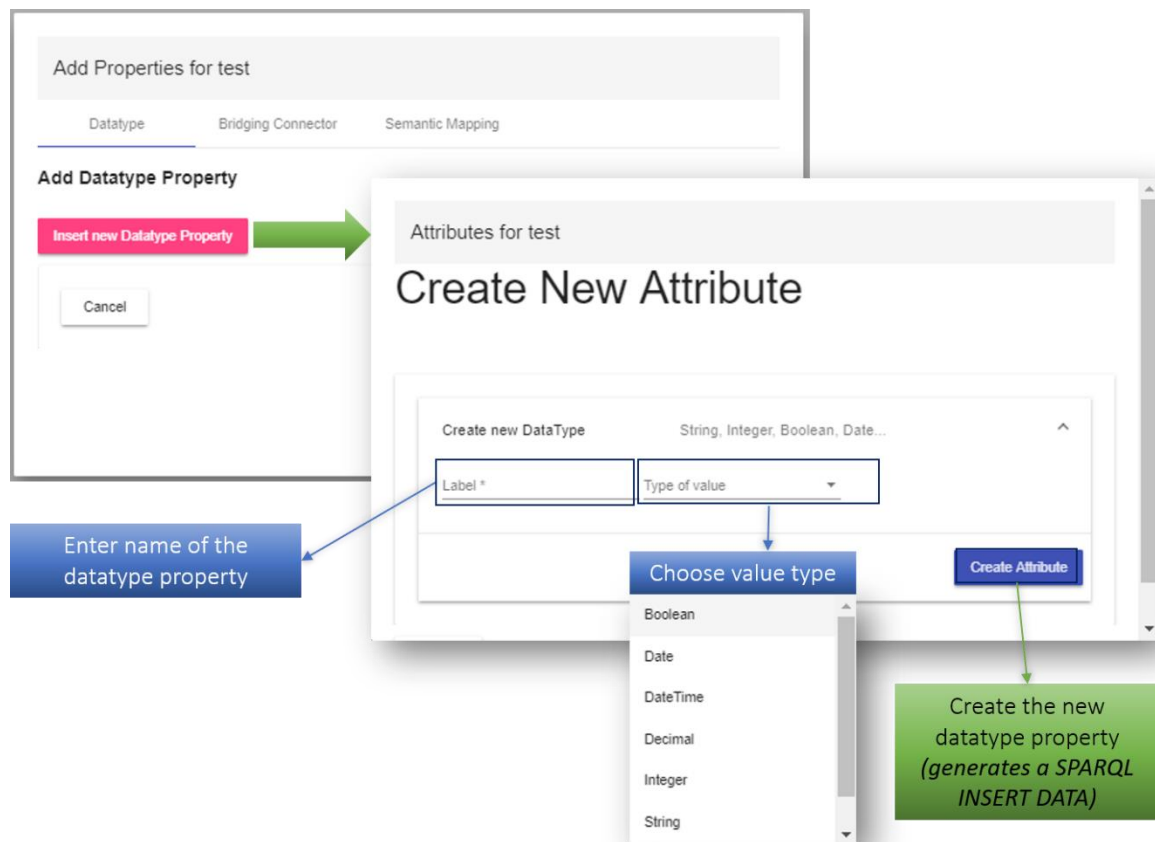


Figure 129. Creating a datatype property

6.3.1.3 Creating New Bridging Connectors (Object Properties)

Bridging connectors are in the form of object properties. The right-bottom side of Figure 130 shows the view, which appears after clicking to the button “Insert new Bridging Connector”. The view foresees the following four fields:

- The label field allows entering the name of the bridging connector.
- The create range field allows selecting a Modelling Language Ontology concept. A search function is implemented in this field for the quick retrieval of the wanted language ontology concept. The search function generates a SPARQL SELECT for each entered character. The query is executed against all the ontology concepts that are sub-classes of *lo:ModellingElement*. The java method can be found under the endpoint */getModellingLanguageOntologyElements*.
- The create object property button allows creating the bridging connector between the newly created modelling construct and an existing language concept. The button leads to generate an instance of **SPARQL Rule 3b**. The java method can be retrieved under the endpoint */createBridgingConnector* in the Java class *ModellingEnvironment.java* of the Web Service.

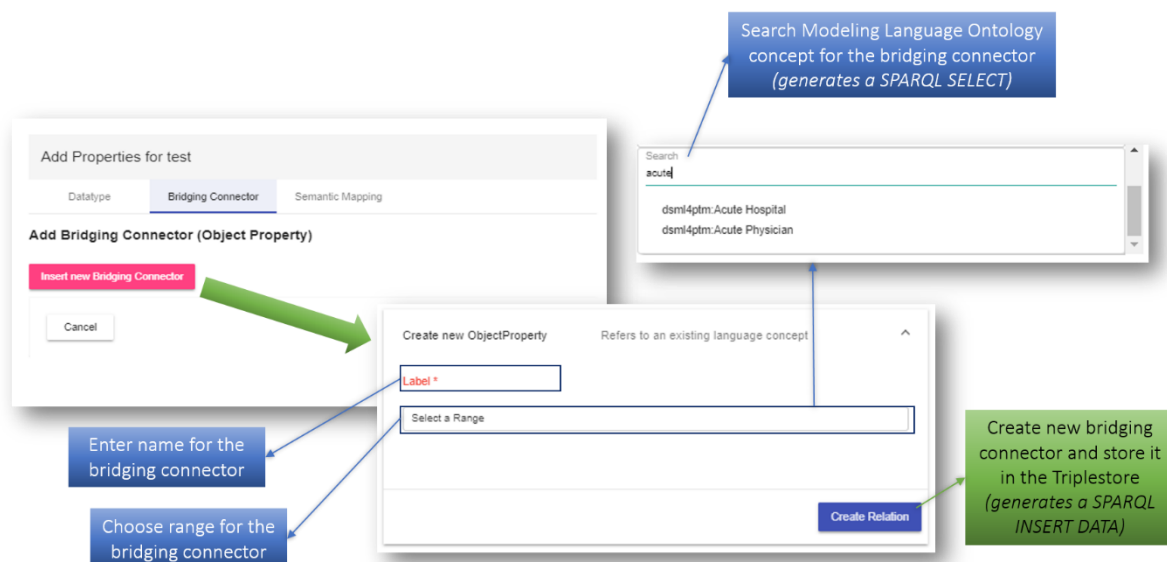


Figure 130. Creating a bridging connector (object property)

6.3.1.4 Creating New Semantic Mappings (Object Properties)

Similar to bridging connectors, semantic mappings are in the form of object properties. The right-bottom side of Figure 131 shows the view, which appears after clicking to the button “Insert new Semantic Mapping”. The view foresees the following four fields:

- The label field allows entering the name of the semantic mapping.
- The range field allows selecting a concept from the Domain Ontology. A search function is implemented in this field for the quick retrieval of the wanted domain ontology concept. The search function generates a SPARQL SELECT for each entered character. The query is fired against all the sub-classes of the *do:DomainOntologyConcept*. The java method can be found under the endpoint */getDomainOntologyClasses*.
- The create object property button allows creating the semantic mapping between the newly created modelling construct and an existing domain concept. The button leads to generate an instance of **SPARQL Rule 3a**. The java method can be retrieved under the endpoint */createSemanticMapping* in the Java class *ModellingEnvironment.java* of the Web Service.
- The create new domain element button leads to the below view shown in Figure 131 and allows creating a new domain element.

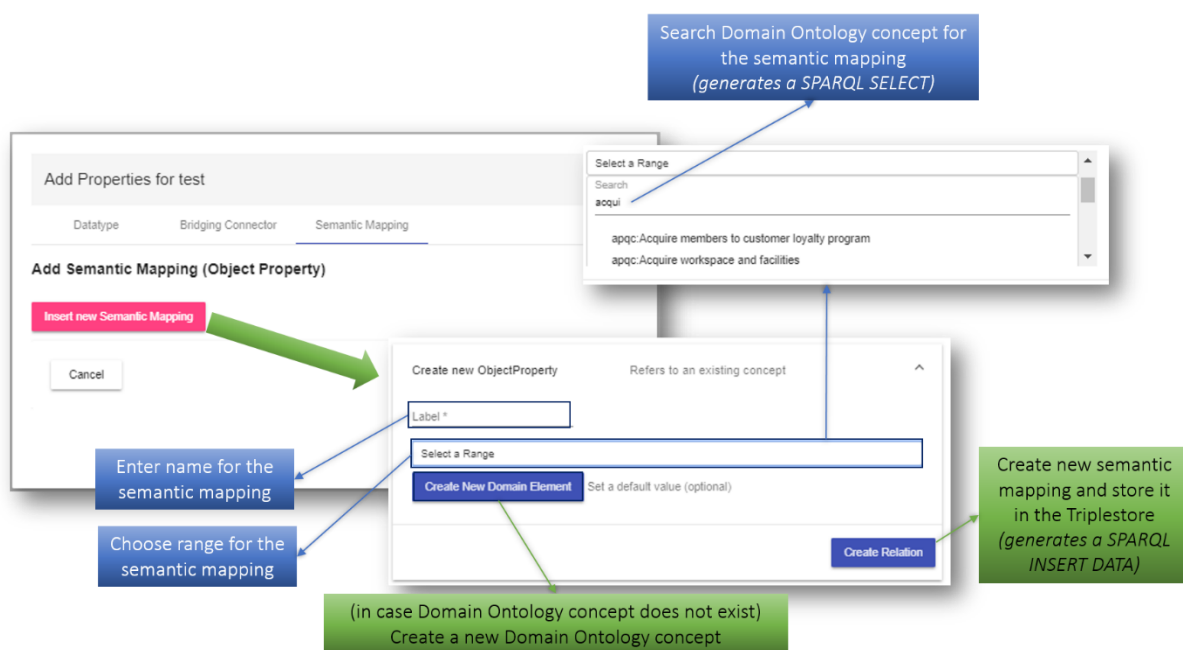


Figure 131. Creating a semantic mapping (object property)

6.3.1.5 Creating New Domain Ontology Concepts

The “Create New Domain Element” button (see Figure 131) leads to the view depicted in Figure 132. This view contains the functionality that allows creating a new concept in the Domain Ontology. The view foresees the following five fields:

- The checkbox root element allows making a domain ontology concept as a root element. That is, if the checkbox is ticked, the new concept will be inserted as a direct sub-class of *do:DomainOntologyConcept*. On the other hand, unticked checkbox enables the selection of a parent class.
- The select parent class field allows selecting a concept from the Domain Ontology. Same as in the creation of the semantic mapping, this field makes use of the Java method for the quick retrieval of the wanted domain ontology concept.
- The label field allows entering the name of the concept.
- The create new domain concept button generates an instance of **SPARQL Rule 5**. The java method can be retrieved under the endpoint *//createDomainElement* in the Java class *ModellingEnvironment.java* of the Web Service. After clicking the button, the view returns to the previous one (see Figure 131).
- The cancel button allows aborting the creation of the domain element and returns to the view depicted in Figure 131.

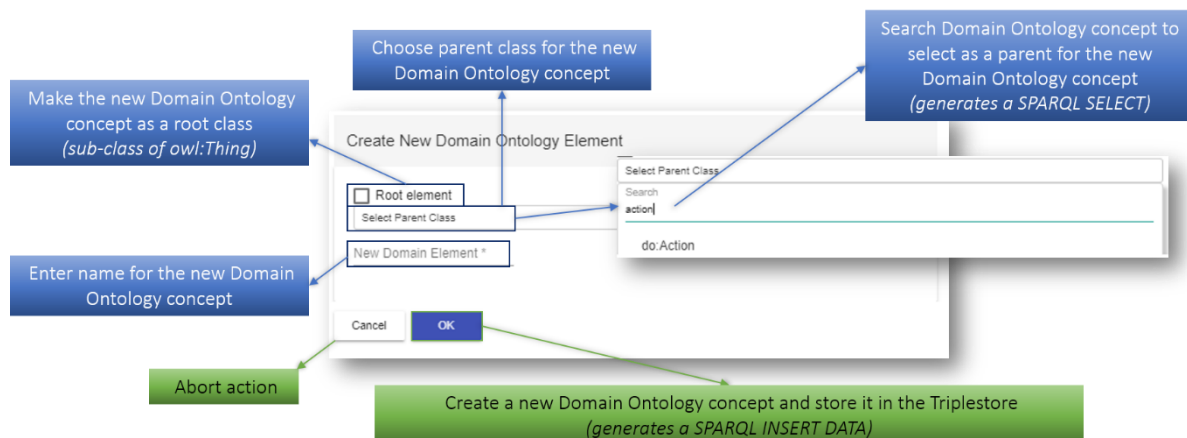


Figure 132. Creating a Domain Ontology concept for the semantic mapping

6.3.1.6 Extending Modelling Constructs via Integration

The view shown in Figure 133 contains the functionality to extend a modelling construct via integration and comprises the following fields:

- The label field shows the name of the selected modelling construct.
- The search field allows selecting a language ontology concept. A search function is implemented to quickly identify the concept to be added as a sub-class. Same as for the bridging connectors, this search function generates a SPARQL SELECT for each entered character and the called java method is also the same */getModellingLanguageOntologyElements*.
- The cancel button allows aborting the integration between modelling constructs and returns to the main view.
- The integration button allows integrating two language ontology concepts. Similar to the previous view, a SPARQL rule (INSERT DATA) is generated by clicking the button, which makes the integration possible (the statement is only generated if a concept is selected from the search field). The generated SPARQL is an instance of **SPARQL Rule 1**. The code of the implemented method can be found under the endpoint */createModellingLanguageSubclasses* in the Java class *ModellingEnvironment.java* of the Web Service.

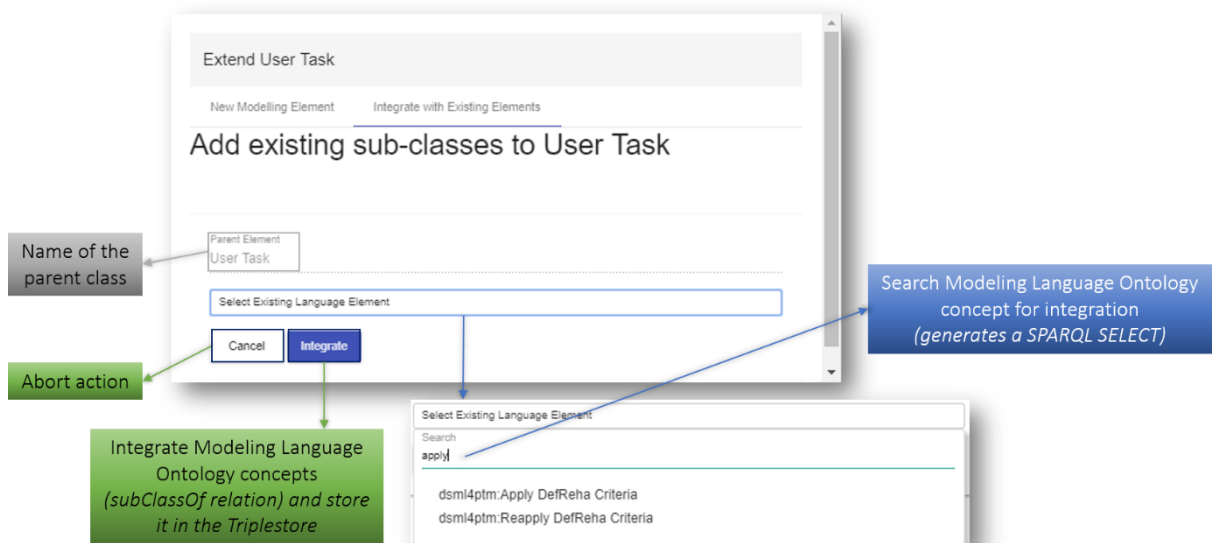


Figure 133. Extending a modelling construct via integration

6.3.2 Feature 2: Editing Modelling Constructs

This sub-section introduces the different functionalities in the GUI that enable editing a modelling construct.

Figure 134 depicts the first screen for editing a modelling construct. The view consists of a pop-up window, which appears after right-clicking on a modelling construct from the palette (see left-hand side of Figure 134). The pop-up shows the fields from the ontology-based meta-model. Similar as for the extension feature, the following field types can be distinguished:

- non-interactive fields, which cannot be edited and have the purpose to be informative (see grey boxes Figure 134). These are the name of the current tab and the ontology prefix, which is bounded to the ontology file, in which the construct resides.
- interactive fields that allow changing knowledge (see blue boxes in Figure 134). These are the label and comment and the changes of graphical notations for palette and Model Editor.
- interactive buttons that lead to a different view (see green box Figure 134). There are the tabs that lead to edit properties: datatype properties, bridging connectors and semantic mappings (see on top of Figure 134).

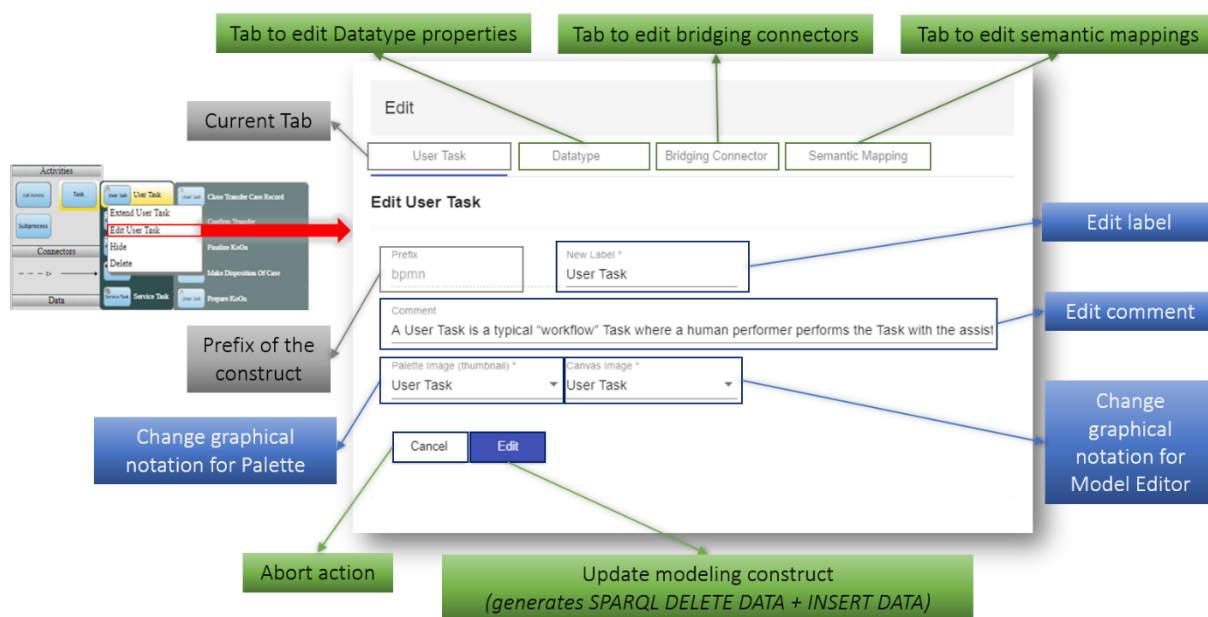


Figure 134. Editing a modelling construct

The edit button updates the modelling construct. The button generates an instance of **SPARQL Rule 8**, i.e. a sequential execution of one SPARQL DELETE DATA and one SPARQL INSERT DATA. The java method can be retrieved under the endpoint */modifyElement* in the Java class *ModellingEnvironment.java* of the Web Service. After clicking the button, the view returns to the main view shown in Figure 125.

Note that the annotation properties that are displayed in the interactive fields (see blue boxes in Figure 134) are retrieved employing a SPARQL SELECT that is dynamically generated by the method *AllPaletteElements()* in the endpoint *getPaletteElements* (see Sub-section 6.2.1.2). Data retrieved from the query are therefore not only used to populate the palette, but also is re-used for displaying the annotation properties of a modelling construct.

6.3.2.1 Editing or Removing Datatype Properties, Bridging Connectors and Semantic Mappings

Editing a modelling construct may include not only the change of (if existing) its datatype property, bridging connector(s) and semantic mapping(s) but also their removal. Therefore, each of the three remaining tabs that are shown on top of Figure 134 leads to edit and delete properties.

6.3.2.1.1 The Datatype Tab

The view for the Datatype tab is depicted on the left-hand side of Figure 136. The fields in the view are the following:

- While the selected tab is “Datatype” in the current view, the four tabs on top of the view remain inactive, as shown in Figure 134.
- The insert new datatype property leads to the view shown in Figure 129.
- Both informative and interactive field for the datatype property allows visualizing all the current datatype properties as well as selecting one. After the selection of one property, the two buttons “Modify” and “Delete” will appear (see grey box named “Selected datatype property for editing” in Figure 136).

Like for the annotation properties, the datatype properties are displayed through a dynamically generated SPARQL SELECT. The query itself is, however, different and generated by the method *queryAllDatatypeProperties(String domainName)* in the endpoint *getDatatypeProperties* (see Figure 135). Each time the Datatype tab is selected, the method is executed, and the input parameter takes the value of the selected modelling construct, which is the domain of the datatype property.

```
@getDatatypeProperties/{domainName}

queryStr.append("SELECT DISTINCT ?id ?domain ?range ?label WHERE {");
queryStr.append("?id a ?type . FILTER(?type IN (owl:DatatypeProperty)) . ");
queryStr.append("?id rdfs:domain ?domain . ");
queryStr.append("FILTER(?domain IN (<" + domainName + ">)) . ");
queryStr.append("?id rdfs:label ?label . ");
queryStr.append("?id rdfs:range ?range . ");
queryStr.append("} ");
queryStr.append("ORDER BY ?label");
```

Figure 135. Excerpt of the method that generates the SPARQL SELECT to retrieve datatype properties

The modify button for the selected datatype property leads to the view on the right-hand side of Figure 136.

The delete button of the selected datatype property generates an instance of **SPARQL Rule 7**. The automatic generation of the statement is allowed by the Java method implemented in the endpoint */deleteDatatypeProperty* in the Web Service.

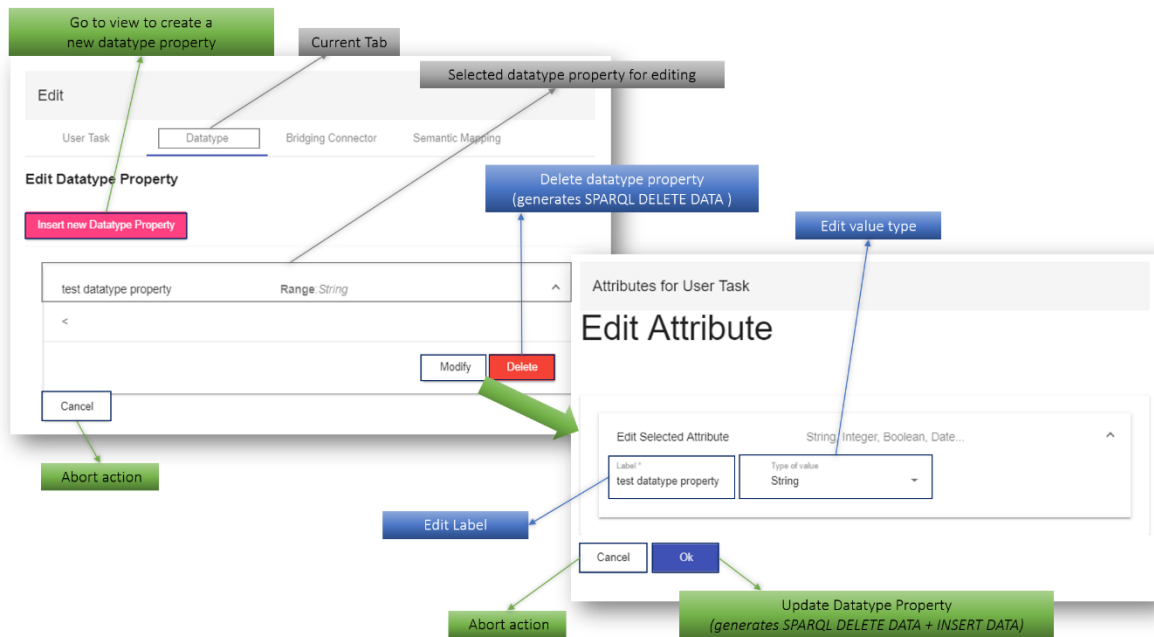


Figure 136. Editing Datatype Property of a modelling construct

In the functionality in the view on the right-hand side of Figure 136 allows editing datatype properties. The view contains the following fields:

- The change name field allows changing the name of the property.
- The change value type field allows changing the value type of the datatype property.
- The cancel button allows interrupting the editing activity for the selected property and returns to the view shown on the left-hand side of Figure 136.
- The apply change button generates an instance of **SPARQL Rule 10**, i.e. a sequential execution of one SPARQL DELETE DATA and one SPARQL INSERT DATA is performed. The dynamic generation of the rule instance is done by the java method implemented in the endpoint */editDatatypeProperty* in the Java class *ModellingEnvironment.java* of the Web Service. After clicking the button, the view returns to the view shown on the left-hand side of Figure 136.

6.3.2.1.2 The Bridging Connector and Semantic Mapping Tabs

Like in the Datatype tab, the two tabs for bridging connectors and semantic mappings are implemented to insert, delete, and change. The look and feel are the same for the three tabs that deal with properties. Due to their similarity, the screenshots for both views bridging connector and the semantic mapping are omitted. Instead, the differences between the datatype property types (see left-hand side of the Figure 136) and the two tabs views of the two object property types are described as follows:

- The insert button to create new bridging connectors or semantic mappings leads to the view depicted in Figure 130 and Figure 131, respectively.
- The existing bridging connector(s) are shown in the tab Bridging Connector, whereas existing semantic mapping(s) are shown in the tab Semantic Mapping. In both cases, the properties are selectable like in the datatype property shown in Figure 136. Figure 137 depicts two excerpts of the two methods that dynamically

generate the SPARQL SELECT. The query retrieves (1) the semantic mapping(s) and (2) the bridging connector(s) of a modelling construct. The methods reside in the endpoints *getSemanticMappings* and *getBridgeConnectors*, respectively.

(1) @getSemanticMappings/{domainName}

```
queryStr.append("SELECT DISTINCT ?id ?domain ?range ?label WHERE {");
queryStr.append("?id rdfs:subPropertyOf ?subProperty . FILTER(?subProperty IN (lo:elementIsMappedWithDOConcept)) . ");
queryStr.append("?id rdfs:domain ?domain . ");
queryStr.append("FILTER(?domain IN (<" + domainName + ">)) . ");
queryStr.append("?id rdfs:label ?label . ");
queryStr.append("?id rdfs:range ?range . ");
queryStr.append("} ");
queryStr.append("ORDER BY ?label");
```

(2) @getBridgeConnectors/{domainName}

```
queryStr.append("SELECT DISTINCT ?id ?domain ?range ?label WHERE {");
queryStr.append("?id rdfs:subPropertyOf ?subProperty . FILTER(?subProperty IN (lo:elementHasBridgingConcept)) . ");
queryStr.append("?id rdfs:domain ?domain . ");
queryStr.append("FILTER(?domain IN (<" + domainName + ">)) . ");
queryStr.append("?id rdfs:label ?label . ");
queryStr.append("?id rdfs:range ?range . ");
queryStr.append("} ");
queryStr.append("ORDER BY ?label");
```

Figure 137. Excerpt of the methods that generate SPARQL SELECT to retrieve (1) semantic mappings and (2) bridging connectors

Same as for the datatype properties, when an object property is selected the two buttons “Modify” and “Delete” appear.

- The delete button generates an instance of **SPARQL Rule 7**. The dynamic generation of the rule is allowed by the Java method implemented in the endpoint */deleteObjectProperty* in the Web Service.

Also, when comparing the “Edit” view of the datatype property (see right-hand side of Figure 136) with the “Edit” views of both bridging connectors and semantic mappings, the only differences are the following:

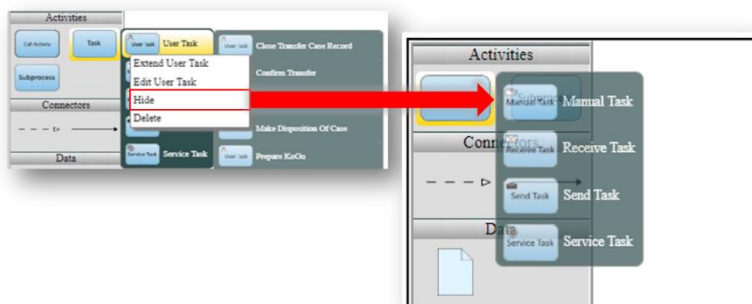
- The change value type fields are replaced by a search field. For the bridging connectors, the implemented search field is shown in Figure 130 whereas for the semantic mappings it is shown in Figure 131.
- In both bridging connectors and semantic mappings, the apply change button generates an instance of **SPARQL Rule 9**. That is, the SPARQL DELETE DATA and then SPARQL INSERT DATA are executed in sequence. The method for the dynamic generation of the rule instance is in the endpoint */editObjectProperty*.

The mechanisms for the rest of the fields such as for the cancel button and change label button are the same as for the datatype property.

6.3.3 Feature 3: Hiding Modelling Constructs

The “Hide Modelling Construct” feature enables to hide the graphical notation of a modelling element or relation from the palette. The hiding can be carried by clicking on the “Hide” button from the context menu. The transition is shown in Figure 138, where on the left-hand side of the figure a graphical notation of User Task is going to be hidden. In result, the right-hand side of Figure 138 shows the list of the graphical notations without User Task.

The “Hide” button leads to generate an instance of **SPARQL Rule 11**. The instance rule changes the value of the datatype property *po:paletteConstructIsHiddenFromPalette* from *False* to *True*. Hence, the graphical notation disappears from the palette. The rule instance is dynamically generated by the Java method in the endpoint */hidePaletteElement* in the Web Service. Figure 139 shows an excerpt of the Java method, in which one can distinguish the part that generates the SPARQL DELETE DATA (upper part of the figure) from the part the generates the SPARQL INSERT DATA (bottom of the figure).



@hidePaletteElement

```
ParameterizedSparqlString querStr = new ParameterizedSparqlString();
querStr.append("DELETE DATA {");
System.out.println("    Element ID: " + pElement.getUuid());
querStr.append("po:" + pElement.getUuid() + " po:paletteConstructIsHiddenFromPalette false;");
querStr.append("}");
```

```
ParameterizedSparqlString querStr1 = new ParameterizedSparqlString();
querStr1.append("INSERT DATA {");
System.out.println("    Element ID: " + pElement.getUuid());
querStr1.append("po:" + pElement.getUuid() + " po:paletteConstructIsHiddenFromPalette true;");
querStr1.append("}");
```

6.3.4 Feature 4: Deleting Modelling Constructs

The “Delete Modelling Construct” feature allows deleting a modelling element or relation.

1. The instance in the Palette Ontology that refers to the selected graphical notation is deleted (e.g. see context menu on the left-hand side of Figure 140).
2. The class in the Modelling Language Ontology that is related to the deleted instance will also be deleted.

The result in the GUI is the same as for the “Hide” feature (shown in Figure 138), where the graphical notation disappears from the palette. Differently from the “Hide” feature, a deleted modelling construct is deleted from the triplestore. The “Delete” feature leads to generate an instance of **SPARQL Rule 6**. The dynamic generation of the rule instance is done through the Java method implemented in the endpoint */deletePaletteElement* in the Web Service.

As described in the critical reflection of the use of operators (see Sub-section 5.1.6) deleting a modelling construct can create inconsistency issues in the knowledge base. For example, a root concept in the abstract syntax might be deleted, which implicates all its sub-concept would be deleted too. In order to avoid that, a method was implemented in the Web Application, which allows deleting only those modelling constructs being leaves in the language ontology taxonomy. Figure 140 depicts the exception that is thrown by the implemented method when the user tries to delete a modelling construct that has at least one sub-concept. The exception is thrown in the form of a pop-up window and alerts the user about the element containing child elements.

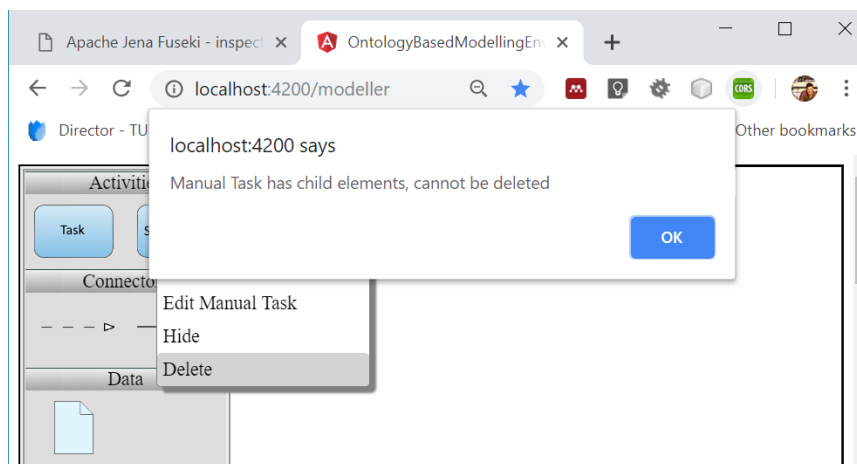


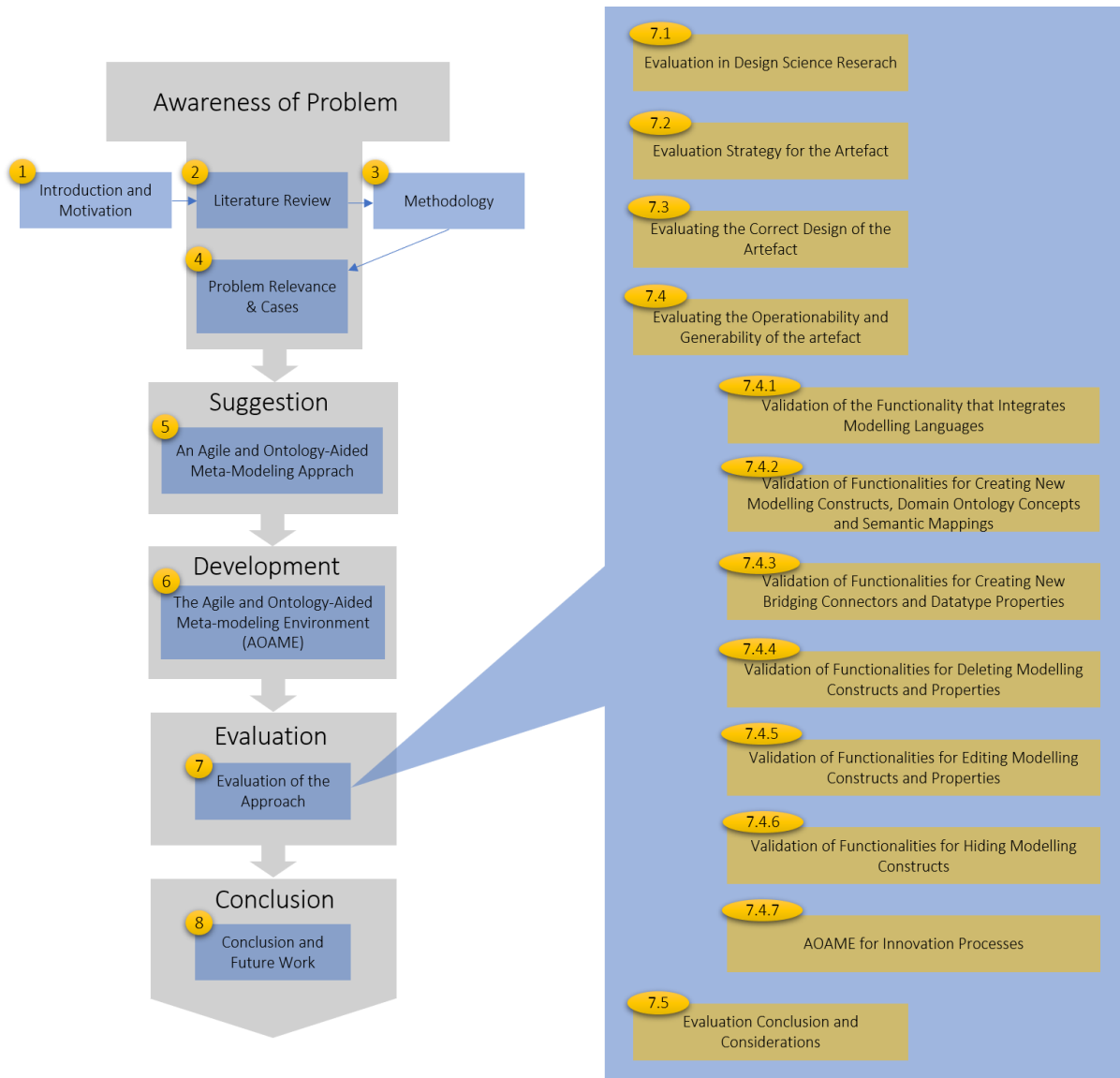
Figure 140. Thrown exception when trying to delete a modelling construct with child elements

6.4 Concluding Summary

In this chapter it was shown how the agile and ontology-aided approach (suggested in Chapter 5) is implemented in a modelling environment called AOAME. The Model-Control-Viewer design pattern was applied to conceive the architecture of AOAME. On the one hand, the architecture supports the propagation from the machine- to the human-interpretable representation of a modelling language. On the other hand, the architecture supports the propagation of domain-specific adaptations from the human- to the machine-interpretable representation of a modelling language. For this, AOAME implements functionalities for the dynamic instantiation of the eleven SPARQL rules, which have been elaborated in Chapter 5.

The automation of the two-way propagation of knowledge allows preserving a seamless consistency between the graphical and machine-interpretable representation of modelling languages. Therefore, the fourth research question is answered.

7. EVALUATION OF THE AGILE AND ONTOLOGY-AIDED META-MODELLING APPROACH



According to March and Storey (2008) “the representation of design problems and the generation and evaluation of design solutions are the major tasks in design science research”.

While the problem design is discussed in Chapter 1, Chapter 2, and Chapter 4, the generation of the design solution is described in Chapter 5.

This chapter deals with the evaluation of the agile and ontology-aided meta-modelling approach; beginning with Section 7.1, and a description of the relevant aspects considered for an evaluation in design science research. Based on these aspects, Section 7.2 describes the strategy followed to evaluate the new approach produced in this research work. Subsequently, Section 7.3 presents an evaluation of the correct design of the approach, and Section 7.4 proceeds to discuss its utility and scope. Results of the evaluation are finally collated in the conclusive section of this chapter (Section 7.5).

7.1 Evaluation in Design Science Research

Evaluation in Design Science Research (DSR) aims to determine the progress achieved by designing, constructing, or using an artefact in relation to the identified problem and the design objectives (March and Smith 1995; Aier and Fischer, 2011; Sonnenberg and vom Brocke, 2012). To achieve this aim, an evaluation strategy shall be built.

The strategic framework proposed by Pries-Heje et al. (2008) helped design science researchers to build strategies for evaluation of their research outcomes and to achieve improved rigour in DSR. The framework is constructed via the analysis of prior DSR evaluation strategies and distinguishes them along three dimensions: (1) *what to evaluate*, (2) *when to evaluate*, (3) *how to evaluate*.

The three dimensions are henceforth explained and contextualised to fit the evaluation strategy for this research work. Further literature is considered in the elaboration of the “when” and “how” dimensions, which provides deeper guidance to build the evaluation strategy.

(1) What to evaluate dimension:

This dimension refers to what is actually evaluated. According to Pries-Heje et al., (2008), either the design process or the design product can be evaluated.

(2) When to evaluate dimension:

This dimension refers to when the evaluation takes place. Pries-Heje et al., (2008) pointed out that “evaluation is not limited to a single activity conducted at the conclusion of a design-construct-evaluate cycle”. Evaluation in information systems can be conducted (1) *ex ante*, which means that artefacts are evaluated prior to their implementation or construction, and (2) *ex post*, where artefacts are evaluated after their design or construction.

The work of Sonnenberg and vom Brocke (2012) further refines the *ex ante* and *ex post* dimensions by proposing evaluation activities after each DSR activity, i.e. problem identification, design, construction and use (see Figure 141). Each evaluation activity (see boxes *Evaln*) contains a feedback loop to the preceding design activity. Feedback loops run in the opposite direction as the DSR cycle. Hevner et al. (2004) consider feedback loops essential to fostering the quality of the design process and the design product under development.

Sonnenberg and vom Brocke (2012) derived evaluation patterns from prior DSR evaluation strategies and refer to each evaluation activity of Figure 141, i.e. *Eval1*, *Eval2*, *Eval3* and *Eval4*. Hence, the identification of the evaluation activity within the time dimension determines the evaluation pattern(s) to adopt.

The evaluation occurring after an artefact implementation corresponds to *Eval3* (see red rectangle on the box *Eval3* in Figure 141)

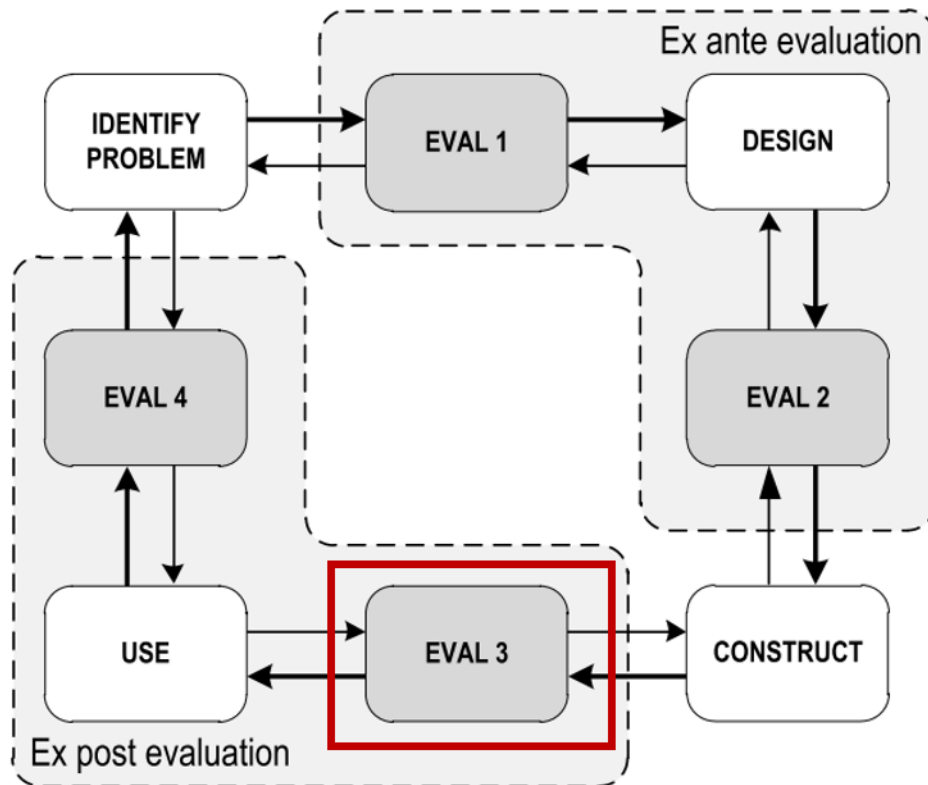


Figure 141. Evaluation activities within a Design Science Research process (Sonnenberg & vom Brocke, 2012)

Eval3 serves to initially demonstrate if and how well the artefact performs to provide a solution to the problem. This ultimately proves the utility of the artefact (Aier & Fischer, 2011).

According to Sonnenberg and vom Brocke (2012), the identified evaluation activity is frequently supported by prototypes. Prototyping aims to demonstrate the utility or suitability of an artefact (Peffer et al., 2012). Cleven, Gubler and Hüner, (2009) add that this type of method facilitates the assessment of a solution's suitability for a certain problem by implementing the solution generically. The evaluation with a prototype represents an adequate evaluation method for design-science research artefacts (March and Storey, 2008).

Since the prototype AOAME instantiates the agile and ontology-aided meta-modelling approach, it is suitable to demonstrate that the approach works in practice and solves the identified problems. A discussion of how to evaluate the artefact through the prototype is presented in the following dimension.

(3) How to evaluate dimension:

The "how to evaluate" dimension refers specifically to how an artefact is evaluated and, (according to Pries-Heje et al. (2008)), should answer the following questions: how is the artefact being evaluated, naturalistically or artificially? What evaluation method is used? and what criteria is it evaluated against? Subsequently, the answer is elaborated upon, by first introducing the differences between artificial and naturalistic evaluation. An evaluation measurement approach is incorporated in this dimension.

(a) *Artificial vs. Naturalistic evaluation form*

According to Pries-Heje et al. (2008) there are two primary forms of evaluation: artificial and naturalistic. The artificial form evaluates an artefact in a contrived and non-realistic way. Sonnenberg and vom Brocke (2012) state that the artificial evaluations refer to evaluation of the artefact against a research gap. The latter is defined originally by Cleven, et al. (2009):

“The artefact is evaluated in respect of correct design either against established or against previously defined requirements. An employment under real world conditions is not realised. The correctness of the derived research gap remains unreflected”.

The naturalistic evaluation, conversely, explores the performance of an artefact by considering aspects of the real-world environment. Such aspects aim to accomplish real tasks in real settings, including evaluation realities such as real users, real systems and real problems (Pries-Heje et al. 2008). Such evaluations, thereby, could incorporate the organisational context both partially or entirely (Sonnenberg and vom Brocke 2012). According to Hevner et al. (2004) the naturalistic evaluations are critical to prove the artefact’s utility for practice. Thus, in contrast to the artificial evaluation, the naturalistic evaluation refers to evaluation of the artefact against the world (Sonnenberg and vom Brocke 2012). The latter is again defined by Cleven, et al. (2009):

“In this case the suitability of the construed artefact is assessed through employment in the real world. Thereby, it becomes obvious if the artefact actually endows utility as a solution for the original problem. The adequacy of the research gap is at the same time implicitly reflected”.

A prototype evaluation method type supports both artificial and naturalistic forms of evaluation (Sonnenberg and vom Brocke 2012).

As an additional input for evaluation through the prototype, the “Illustrative Scenario” method can be considered. This method type is defined by Peffers et al. (2012) as the “application of an artefact to a synthetic or real-world situation aimed at illustrating suitability or utility of the artefact”. Both prototype and illustrative scenario have the same purpose (i.e. evaluating utility of the artefact) and can support both artificial and naturalistic forms of evaluation.

The utility of the artefact and correct design are both *Ex Post* as shown in Table 54.

Table 54. Ex post artificial and naturalistic evaluation strategy for the artefact. Adapted from (Pries-Heje et al. 2008)

	Ex Ante	Ex Post
Naturalistic		Utility of the artefact (support of the Prototype and the Illustrative Scenario method types)
Artificial		Correct design of the artefact (support of the Prototype method type)

(b) Evaluation Criteria and Artefact Type

To systematically show the progress that an artefact aims at, the evaluations should be guided by evaluation criteria (Aier & Fischer, 2011).

March and Smith (1995) suggest that the appropriate derivation of the evaluation criteria depends on the artefact type that has been produced, i.e. construct, model, method or instantiation. For instance, among others, there are completeness, simplicity and elegance for the construct artefact type whereas fidelity with real world phenomena, level of detail, robustness, and internal consistency for model artefact type. Table 55 shows the suggested evaluation criteria for any given artefact type.

Table 55. Evaluation criteria for Design Science Research artefact types (March and Smith, 1995)

	Construct	Model	Method	Instantiation
Completeness	X	X		
Ease of use	X		X	
Effectiveness				X
Efficiency			X	X
Elegance	X			
Fidelity with real world phenomena		X		
Generality			X	
Impact on the environment and on the artefact's users				X
Internal consistency		X		
Level of detail		X		
Operationality			X	
Robustness		X		
Simplicity	X			
Understandability	X			

A method artefact type represents a set of steps (algorithms, proceedings or guidelines) used for the solution of specific problems or classes of problems, e.g. approaches for business process modelling or software development (Cleven et al., 2009). March and Smith (1995) further specify the concept of methods with system development method:

“System development methods facilitate the construction of a representation of user needs. [...] They further facilitate the transformation of user needs into system requirements [...] and then into system specifications, [...] and finally into an implementation”.

The artefact built in this research fits the definition attributed to a system development method. Given the possibility of the approach to adapt modelling languages on-the-fly, this artefact facilitates the construction of a representation of a user needs. The user needs are accommodated in modelling languages and the ontology-aided approach ensures that they are specified in a machine-interpretable form.

The definitions of other artefact types are not provided as they do not suite the artefact produced in this research.

Possible evaluation criteria for methods are operationality, efficiency, generality, and ease of use (see red squares in Table 55).

With respect to the above-introduced evaluation patterns of Sonnenberg and vom Brocke (2012), the evaluation criteria operationality and generality fit *Eval3* (see red rectangle in Figure 141), which occur after the implementation of the artefact. Namely, operationality is an aspect of utility (see Aier and Fischer, 2011) and concerns the ability to perform the intended task with the method (March and Smith 1995). Generality refers to the purpose and scope of an artefact (Aier & Fischer, 2011), and can be defined as the ability of the method to be applied in different application scenarios or other contexts.

Conversely, the evaluation criteria efficiency and ease of use, are considered in a later stage when analysts apply a method (Sonnenberg and vom Brocke 2012), traditionally with general use of the implemented artefact (see *Eval4* in Figure 141).

The measurement of an artefact with respect to operationality and generality can be achieved through a qualitative approach. According to Chen and Hirschheim (2004), the qualitative evaluation approach “emphasises the description and understanding of the situation behind the factors”, thus characteristics of the evaluation criteria are not appraised on a numerical basis but on a value basis (Cleven et al., 2009). A quantitative approach, on the other hand, would instead fit the measurement with respects to efficiency and ease of use.

7.2 Evaluation Strategy for the Artefact

This section proposes an evaluation strategy suitable for this work. For this, findings considered in the previous section are considered. Table 56 summarizes the evaluation strategy. Each dimension “what”, “when” and “how” is presented with the respective chosen characteristic value and are subsequently described.

In this work the design product is evaluated, i.e. the agile and ontology-aided meta-modelling approach (*what to evaluate dimension*). The evaluation of the approach occurs after its implementation, *Eval3* (*where to evaluate dimension*). As shown in Table 56, the *how to evaluate dimension* contains six sub-dimensions. The prototype method type is used for both an artificial and a naturalistic evaluation. The artificial evaluation form aims to evaluate the correct design of the approach against the previously defined requirements. The naturalistic form is used to evaluate the utility of the approach. Additionally, the illustrative scenario method type is considered for the evaluation of the utility of the artefact. Namely, real-world use cases are proposed to be implemented in the prototype. The considered evaluation criteria are operationality and generality, which are both for the purpose of utility and are consistent to what is shown in Table 54 of Section 7. Hence, the two evaluation criteria fit both the artefact type “method” and the evaluation activity chosen in the “where dimension”: *Eval3*. The two evaluation criteria are thus specified and contextualised to fit this research work:

- Operationability of the approach: The ability of the approach to preserve consistency between the graphical and the machine-interpretable representation while performing on-the-fly domain-specific adaptations of modelling languages.
- Generality of the approach: The ability of the approach to be applied in different application domains.

To evaluate the correct design of the approach against the requirements the criteria is contextualised as follows:

- The extent to which the requirements are satisfied by the implemented approach AOAME.

Finally, a qualitative approach is adopted to measure the artefact with respect to the proposed evaluation criteria. Hence, understanding about the extent to which operationability, generality and design requirements are satisfied and descriptively emphasised.

Table 56. Evaluation Strategy for the Agile and Ontology-Aided Meta-modelling Approach

Dimension		Characteristic values		
What to evaluate		The agile and ontology-aided meta-modelling approach		
When to evaluate		Ex post (<i>Eval3</i>)		
How to evaluate	Evaluation Form	Naturalistic		Artificial
	Evaluation Method	Prototype and Illustrative Scenario		Prototype
	Evaluation purpose	Utility of the artefact	Purpose and scope	Correct design of the artefact
	Artefact type	Method		
	Evaluation criteria	Operationability of the approach	Generality of the approach	Extent to which the requirements are satisfied by AOAME
	Measurement Evaluation Approach	Qualitative		

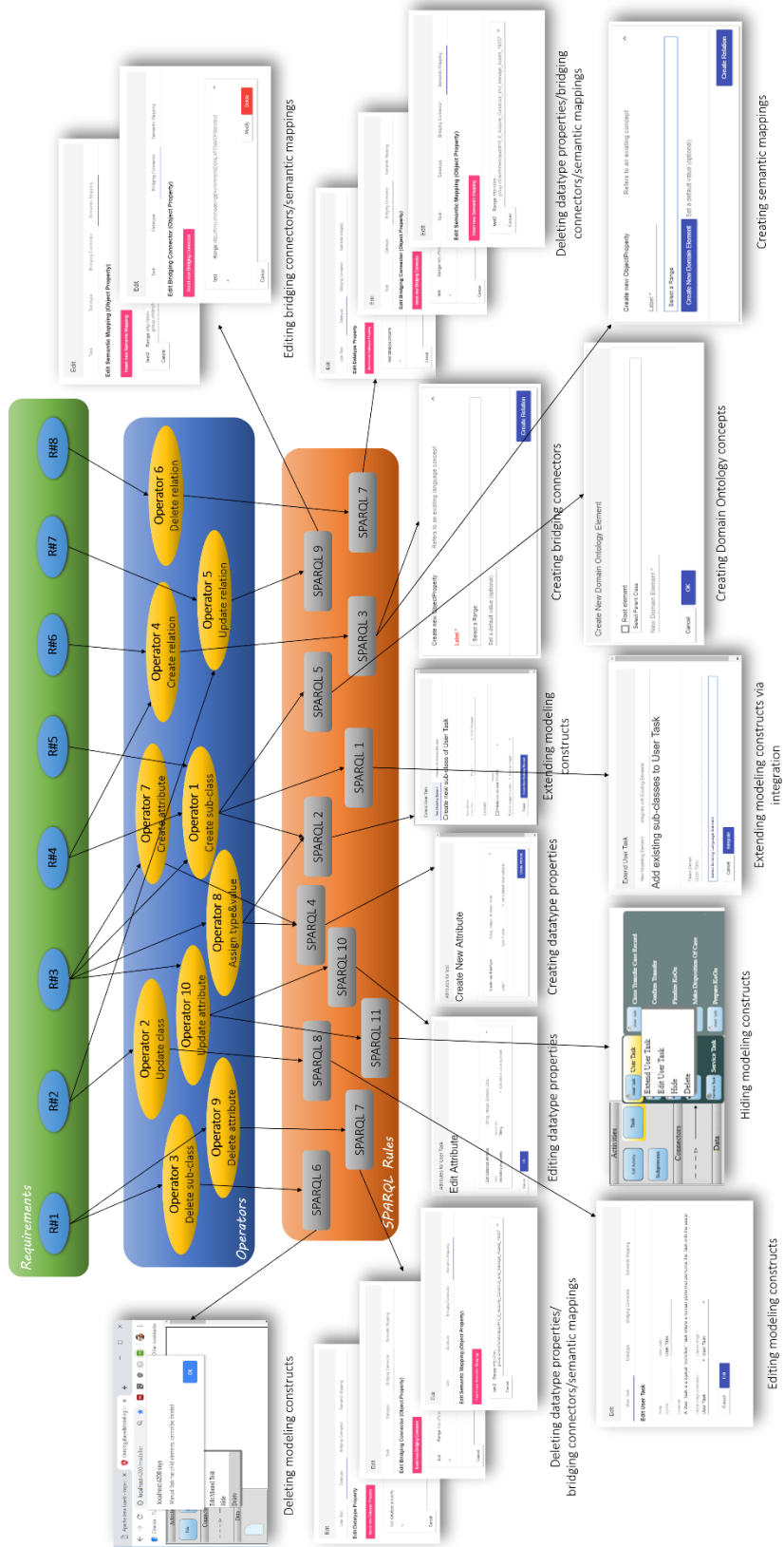
The following two sub-sections elaborate on the actual evaluation of the artefact with respect to (1) the correct design and (2) the utility purposes.

7.3 Evaluating the Correct Design of the Artefact

This section describes the evaluation of the correct design of the agile and ontology-aided meta-modelling approach. As introduced in the evaluation strategy, this evaluation is done against previously defined requirements. Requirements can be found in Section 4.4.5 and Section 5.1.4 and are derived from both literature and interviews with industry experts. The AOAME prototype is used to support this evaluation. As described in the previous Section, the evaluation criteria to consider are the following:

- The extent to which requirements are satisfied by the implemented approach AOAME.

Below, the evaluation is presented by elaborating on the basis of each of the eight requirements. The description is underpinned by Figure 142, which provides a traceable map between the requirements (see top of the figure) and the implemented functionalities in AOAME (see bottom of the figure). The different layers from top to bottom reflect the gradual steps adopted in this research work. After the elicitation of eight requirements, ten operators (see Section 5.1.6) were conceived, accordingly (see mapping between requirements and operators. Subsequently, each operator was implemented by at least one SPARQL rule (see Section 5.3.3). The latter allows propagating domain-specific adaptations from the graphical to the machine-interpretable representation of the modelling language.



Requirement #1: *An agile meta-modelling approach should enable the language engineer to simplify a modelling language.*

Requirement one is satisfied by functionalities of Feature 2 “Editing Modelling Constructs” (Sub-section 6.3.2), Feature 3 “Hiding Modelling Construct” (Sub-section 6.3.3) and Feature 4 “Deleting Modelling Construct” (Sub-section 0). The functionalities of each feature are detailed below.

- *From Feature 2:* “Editing/Removing Datatype Properties, Bridging Connectors, Semantic Mappings” (Sub-section 6.3.2.1) incorporates **SPARQL Rule 7**, which allows the removal of both object and datatype properties. SPARQL Rule 7 supports **Operator 9** - delete attribute and **Operator 6** – delete relation. Both rules update the Modelling Language Ontology.
- *From Feature 3:* “Hiding Modelling Construct” (Sub-section 6.3.3) incorporates **SPARQL Rule 11**, which allows hiding a modelling construct by changing the dedicated datatype property. SPARQL Rule 11 supports **Operator 10** - update attribute. The rule updates the Palette Ontology.
- *From Feature 4:* “Deleting Modelling Construct” (Sub-section 0) incorporates **SPARQL Rule 6**, which allows the removal of modelling constructs. SPARQL Rule 6 supports **Operator 3** - delete sub-class. The rule updates both the Modelling Language Ontology and the Palette Ontology.

Requirement #2: *An agile meta-modelling approach should enable the language engineer to change abstract syntax and notation.*

Requirement two is satisfied by functionalities of Feature 2 “Editing Modelling Construct” (Sub-section 6.3.2). The functionalities are the following:

- “Editing Modelling Construct” (Sub-section 6.3.2) incorporates **SPARQL rules 8**, which allows the label, comment and graphical notation of a modelling construct to be updated. SPARQL rules 8 supports **Operator 2** - update Class. Both the Modelling Language Ontology and the Palette Ontology are updated.
- “Editing/Removing Datatype Properties, Bridging Connectors, Semantic Mappings” (Sub-section 6.3.2.1) incorporates:
 - **SPARQL Rule 10**, which allows the update of datatype properties and supports **Operator 10** - update attribute.
 - **SPARQL Rule 9**, which allows the update of object properties and supports **Operator 5** - update relation.

Both rules update the Modelling Language Ontology.

***Requirement #3:** An agile meta-modelling approach should enable the language engineer to extend abstract syntax and add new notation.*

Requirement three is satisfied by functionalities of Feature 1 “Extending Modelling Construct” (Sub-section 6.3.1). The functionalities are the following:

- “Extending Modelling Construct” (Sub-section 6.3.1) incorporates **SPARQL Rule 2**, which allows the creation of new modelling constructs as a specialization of existing ones. The new modelling construct includes a new name (i.e. Uniform Resource Identifier), label, comment and graphical notation. SPARQL Rule 2 supports **Operator 1** – create sub-class and **Operator 8** – Assign concept, attribute type or value. Both the Modelling Language Ontology and the Palette Ontology are updated.
- “Creating New Datatype Properties” (Sub-section 6.3.1.2) incorporates **SPARQL Rule 4**, which allows the creation of new datatype properties for a modelling construct. SPARQL Rule 4 supports **Operator 7** – create attribute. The Modelling Language Ontology is updated.

***Requirement #4:** An agile meta-modelling approach should enable the language engineer to integrate concepts that belong to different modelling languages or different modelling views.*

Requirement four is satisfied by functionalities of Feature 1 “Extending Modelling Construct” (Sub-section 6.3.1). Namely:

- “Extending a Modelling Construct via Integration” incorporates **SPARQL Rule 1**, which allows adding the sub-class of a relationship between two existing modelling constructs. SPARQL Rule 1 supports **Operator 1** – create sub-class (see Section 6.3.1.6). Both the Modelling Language Ontology and the Palette Ontology are updated.
- “Creating New Bridging Connectors” (Sub-section 6.3.1.3) incorporates **SPARQL Rule 3b**, which allows the creation of bridging connectors between modelling constructs. The latter can belong to different modelling views. SPARQL Rule 3b supports **Operator 4** – create relation. The Modelling Language Ontology is updated.

***Requirement #5:** An agile meta-modelling approach should enable the language engineer to create new semantic domain concepts.*

Requirement five is satisfied a functionality of Feature 1 “Extending Modelling Construct” (Sub-section 6.3.1). The functionality is the following:

- “Creating New Domain Ontology Concepts” (Sub-section 6.3.1.5) incorporates **SPARQL Rule 5**, which allows to create new concepts in the Domain Ontology. SPARQL Rule 5 supports **Operator 1** – create sub-class. The Domain Ontology is updated only.

Requirement #6: *An agile meta-modelling approach should enable the language engineer to create new semantic mappings between concepts from an abstract syntax (linguistic view) to a semantic domain (domain view).*

Requirement six is satisfied by a functionality of Feature 1 “Extending Modelling Construct” (Sub-section 6.3.1). Namely:

- “Creating New Semantic Mappings” **SPARQL Rule 3a**, which allows the creation of new object properties for the semantic mappings between concepts from the Modelling Language Ontology to the Domain Ontology. **Operator 4** – create relation (see views in Section 6.3.1.4). The Modelling Language Ontology is updated.

Requirement #7: *An agile meta-modelling approach should enable the language engineer to modify the semantic mapping between concepts from an abstract syntax (linguistic view) to a semantic domain (domain view).*

Requirement seven is satisfied by a functionality of Feature 2 “Editing Modelling Construct” (Sub-section 6.3.2). Namely:

- “Editing/Removing Datatype Properties, Bridging Connectors, Semantic Mappings” (Sub-section 6.3.2.1) incorporates **SPARQL Rule 9**, which allows the semantic mappings to be changed. SPARQL Rule 9 supports **Operator 5** - update relation. The Modelling Language Ontology is updated.

Requirement #8: *An agile meta-modelling approach should enable the language engineer to delete the semantic mappings between concepts from an abstract syntax (linguistic view) to a semantic domain (domain view).*

Requirement eight is satisfied by a functionality of Feature 2 “Editing Modelling Construct” (Sub-section 6.3.2), namely:

- “Editing/Removing Datatype Properties, Bridging Connectors, Semantic Mappings” (Sub-section 6.3.2.1) incorporates **SPARQL Rule 7**, which allows the removal of semantic mappings. SPARQL Rule 7 supports **Operator 6** - delete relation. The Modelling Language Ontology is updated.

7.4 Evaluating the Operationability and Generality of the Artefact

This section describes the evaluation of the agile and ontology-aided meta-modelling approach with respect to the following criteria:

- *Operationability of the approach*: The ability of the approach to preserve consistency between the graphical and the machine interpretable representation while performing on-the-fly domain-specific adaptations of modelling languages.
- *Generality of the approach*: The ability of the approach to be applied in different application domains.

For the evaluation activity, six use cases were implemented through the prototype AOAME. The use cases are chosen based on two criteria: (1) *Validation of all the AOAME's functionalities*; and (2) *Significance in real-world applications*. The two criteria are detailed below.

1. *Validation of all the AOAME's functionalities*: The validation of all functionalities proves the *operationability* of the agile and ontology-aided meta-modelling approach. All AOAME's functionalities are introduced in Section 6.3. The centre of Figure 143 depicts the six use cases and the used functionalities are on their right- and left-hand side. The arrows indicate which functionality is used in which use case. More specifically:
 - *“Use Case 1: Adding Discretionary Task”* is implemented through the functionality *“Extending Modelling Constructs via Integration”* (see Sub-section 7.4.1).
 - *“Use Case 2: Extending BPMN Group”* (including properties) is implemented through the functionalities *“Creating New Modelling Construct, Domain Ontology Concepts and Semantic mappings”* (see Sub-section 7.4.2).
 - *“Use Case 3: Adding ICF Standard document”* is implemented through the functionalities *“Creating New Bridging Connectors and Datatype Properties”* (see Sub-section 7.4.3).
 - *“Use Case 4: Deleting ICF Standard document and/or properties”* (i.e. annotation properties, datatype properties, bridging connector, semantic mappings) are implemented through the functionalities *“Deleting Modelling Constructs and Properties”* (see Sub-section 7.4.4).
 - *“Use case 5: Editing properties of ICF Standard document”* (i.e. annotation properties, datatype properties, bridging connector, semantic mappings) is implemented through the functionality *“Editing Modelling Constructs and Properties”* (see Sub-section 7.4.5).
 - *“Hiding ICF Standard document”* is implemented through the functionality *“Hiding Modelling constructs”* (see Sub-section 6.3.3).

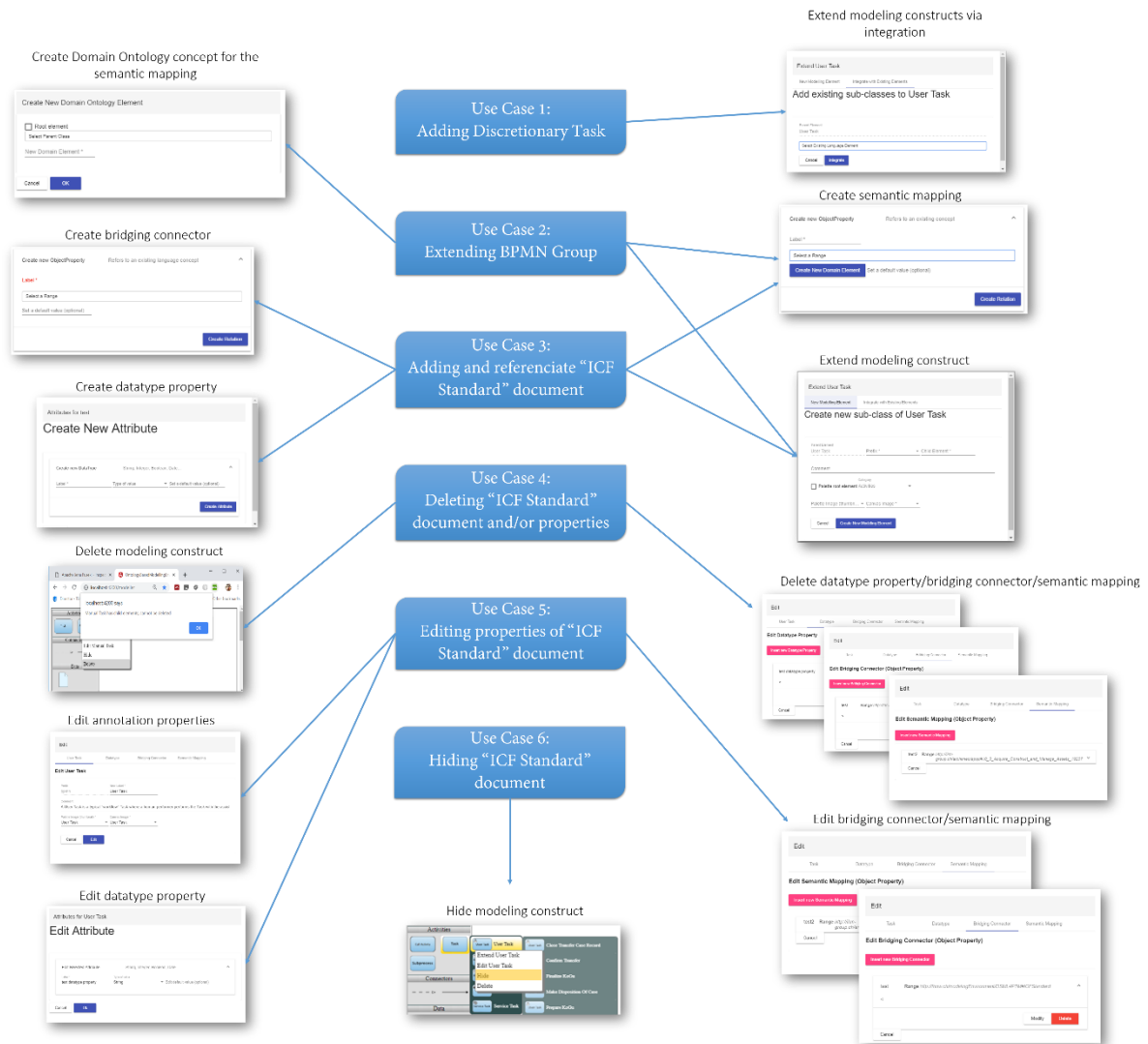


Figure 143. Use cases implemented through AOAME's functionalities

2. *Significance in real-world applications:* Use cases 1 and 3 fall within the issue of process documentation, which is relevant to companies to guide the work of the people involved. According to the Camunda founders Freund and Rucker (2016), process documentation is one of the most requested consulting cases by companies. Use Case 2 does not only deal with process documentation but also allows for ontology-based automation, which can support the decision making of workers. Ontology-based automation (also called Semantic or Explainable Artificial Intelligence) is increasing demand from companies that adopt Artificial Intelligence-based recommendation systems to explain how certain results are derived (see Deloitte, 2019). The rest of the use cases build on Use Case 3 to validate the rest of the functionalities. All use cases were extracted from one of the two application domains Patient Transferal Management (Section 4) and Business Process as a Service (Section 4.2). As introduced in the evaluation strategy (Section 7.1), the pertinence of use cases to the real-world settings ensures the naturalistic evaluation form of the approach and proves the artefact's utility for practice.

To prove *generality* of the approach, firstly, it is demonstrated that AOAME supports the implementation of use cases from two different application domains: Patient Transferal

Management, and Business Process as a Service. Both application domains are introduced in Section 4 and Section 4.2 , respectively. Secondly, it is demonstrated that AOAME supports an additional application domain, which was not previously introduced, concerning with innovation processes (see Sub-section 7.4.7).

The use cases were implemented by me. The implementation included the creation of:

- A set of ontologies: this set consists of the main three ontologies; Modelling Language Ontology, Palette Ontology and Domain Ontology as well as the ontologies created for the use cases: APQC, BPaaS, BPMN, CMMN, DKMM, DSML4PTM, FDPDO, ICF, OrganisationalModel, SAPScenesOntologies. These ontologies will be explained along with the description of the specific use case. At present, all of these ontologies count more than 34'648 triples. The ontologies were engineered in the ontology editor tool TopBraid and can be found in Appendix E: Prototype , folder E4.
- A set of graphical notations: more than 700 graphical notations, including thumbnails for the palette and images for the Modelling Editor were created and can be found in Appendix E: Prototype , folder E5.
- A set of SPARQL queries: nine SPARQL rules were created to prove that the adapted modelling language is consistent with the ontology. The execution of the SPARQL queries was performed in the Fuseki Triplestore, which contains the set of ontologies. To demonstrate that the changes are automatically propagated, the query was executed both before and after the domain-specific adaptations.

The sequence of activities performed for implementing each use case was as follows:

1. Loading the set of ontologies that is required by the use case into the Fuseki Triplestore;
2. Loading (or re-loading) the graphical user interface (GUI);
3. Selecting modelling language in the GUI;
4. Selecting the modelling view of the previously selected modelling language in the GUI;
5. Using AOAME's functionalities for the on-the-fly domain-specific adaptations of modelling languages in the GUI.
6. Executing the SPARQL queries in Fuseki Triplestore.

Finally, each use case is below described (Sub-sections from 7.4.1 to 0) with the following structure:

- Description of the use case and conceptual solution expected from the applied domain-specific adaptations.
- The set of ontologies, which are used to implement the considered use case;
- A set of user actions for the domain-specific adaptations of modelling languages;
- The SPARQL query results, in order to prove that the adapted graphical language is consistent with the ontology.

7.4.1 Validation of the Functionality that Integrates Modelling Languages

This sub-section describes the validation of the AOAME’s functionality “Extending Modelling Constructs via Integration”. For this purpose, the use case “Adding Discretionary Task” is extracted from the case Patient Transferal Management. In the use case, it is shown the manner a modelling element like Discretionary Task from CMMN is used to extend another modelling element from a different modelling language, i.e. Manual Task from BPMN. The use case is also mentioned in Sub-section 0 and published in Laurenzi et al. (2017).

7.4.1.1 Description of Use Case 1: Adding Discretionary Task

In this use case the transferal manager (i.e. domain expert) uses BPMN to model that an acute physician initiates the task “Perform Rehab Conference”. As this task is discretionary, he/she re-uses the modelling element “Discretionary Task”, which is defined in CMMN (OMG, 2016a). As discretionary tasks are always executed by a person, it should be represented as a sub-concept of “Manual Task”, which already exists in BPMN. The integration of modelling elements of one modelling language (in this case the discretionary task), will result in the possibility for the transferal manager to model a discretionary task in a BPMN lane. Figure 144 depicts one excerpt of a DSML4PTM process model containing the discretionary “Perform Rehab Conference” in the BPMN lane “Acute Physician”. The lane represents the role of a performer that initiates the discretionary task “Perform Rehab Conference”.

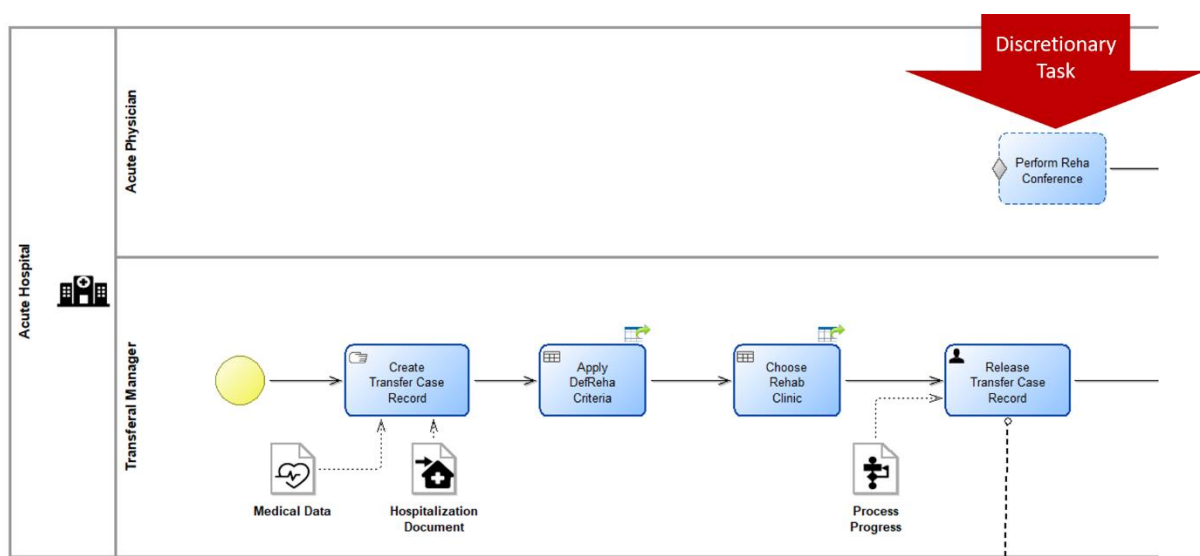


Figure 144. DSML4PTM process model containing a Discretionary Task in a BPMN Lane

The integration of Discretionary Task from CMMN with Manual Task from BPMN is achieved by instantiating SPARQL Rule 1 (see Sub-section 5.3.4.1.2). As a result, the following two following two relations created (see also Figure 145):

- a *hasParent* relation is created between two instances *po:DiscretionaryTask* and *po:ManualTask* of the class *po:PaletteElement*;
- an *rdfs:subClassOf* relation is created between two modelling elements *bpmn:ManualTask* and *cmmn:DiscretionaryTask*.

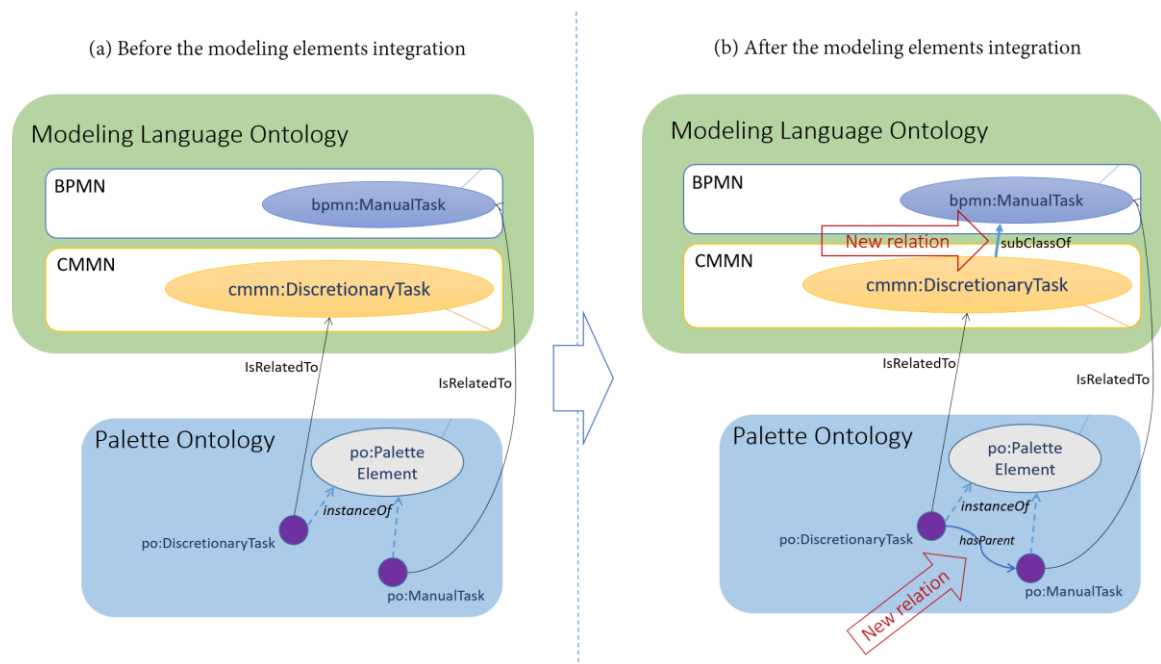


Figure 145. Conceptual solution (a) before and (b) after adding Discretionary Task to Manual Task

7.4.1.2 Set of Ontologies for Use Case 1

The set of ontologies required to implement Use Case 1 are as follows:

- The BPMN ontology: an ontology that reflects the linguistic view of BPMN, and thus containing a class hierarchy, attributes and relations;
- The CMMN ontology: an ontology that reflects the linguistic view of CMMN, and thus containing a class hierarchy, attributes and relations;
- The Modelling Language Ontology (see Sub-section 5.3.2.1). Both ontologies BPMN and CMMN are imported in the Modelling Language Ontology. Specifically, modelling elements from both BPMN and CMMN are entered as sub-classes of *lo:ModellingElement* while modelling relations are entered as sub-classes of *lo:ModellingRelation*. The top-right green corner of Figure 146 shows an excerpt of the class hierarchy of BPMN integrated in the Modelling Language Ontology.
- The Palette Ontology, which is described in Section 5.3.2.1. In this case, the two classes *po:PaletteConnector* and *po:PaletteElement* contain the instances for displaying the graphical notations of BPMN and CMMN. Figure 146 shows an excerpt of the Palette Ontology. Note the relation between the *po* instances and classes in the Modelling Language Ontology should already exist. For example, see connection between *po:Task* and *bpmn:Task* in Figure 146.

These set of ontologies should be loaded to the triplestore (see left-hand side of Figure 147), in order for the palette to be populated (see right-hand side of Figure 147). The palette shows the graphical elements of BPMN 2.0, which was the selected modelling language from the GUI (see top of the palette in Figure 147). The approach that populates the palette has been described in Section 6.2.1.

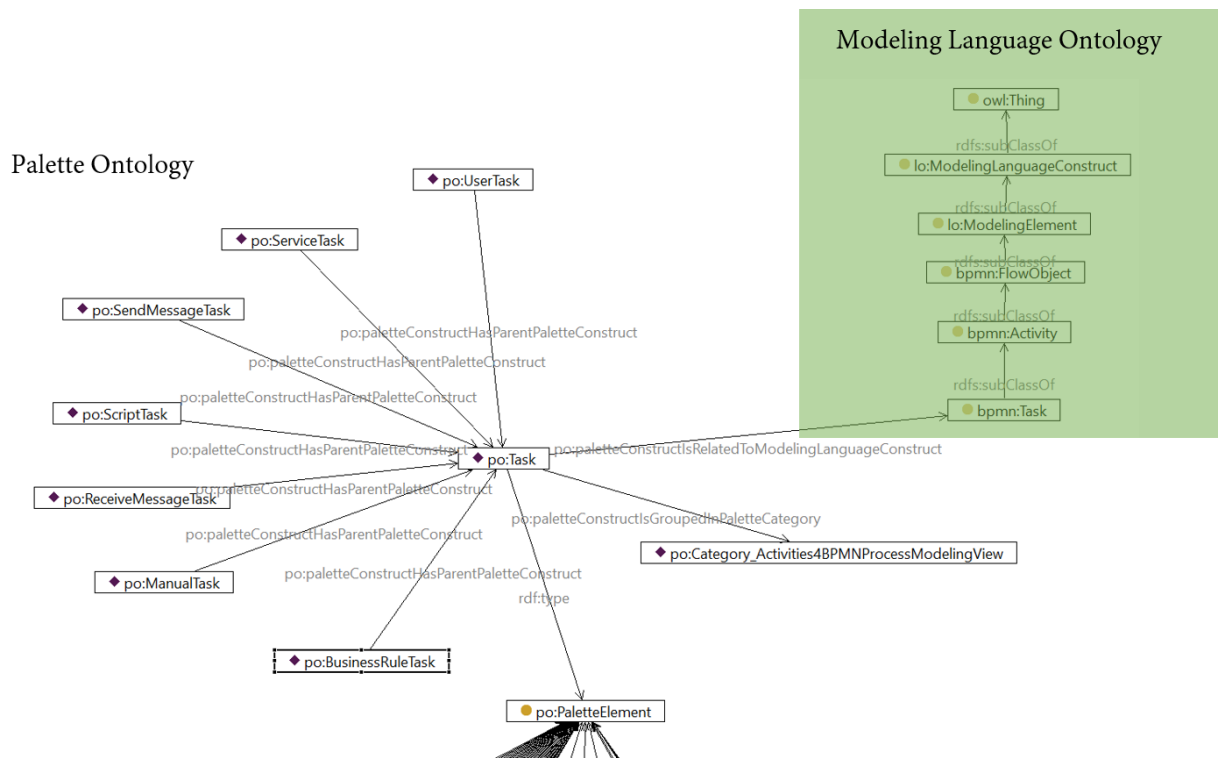


Figure 146. Excerpt of the Palette Ontology and Modelling Language Ontology related to BPMN 2.0

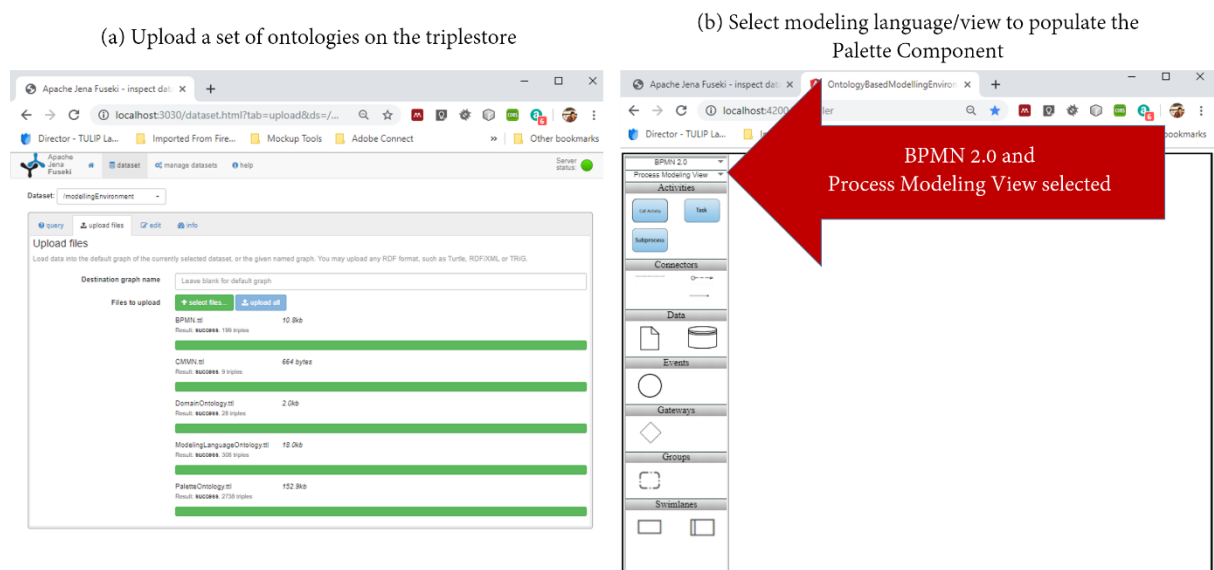


Figure 147. Populating the palette with graphical notations from the “Process Modelling View” of BPMN 2.0

7.4.1.3 User Actions Performed in the Graphical User Interface

Figure 148 underpins the description of the user actions for extending Manual Task with Discretionary Task. The user actions are presented in the following steps:

- Step 1: the user right-clicks on the modelling construct to extend “Manual Task”.

- Step 2: the pop-up window for the extension appears (see second view of Figure 148). After selecting the tab “Integrate with Existing Elements”, the user looks for the concept *cmnn:DiscretionaryTask* by typing the name in the search box. The latter implements the search functionality for querying the ontology. The search functionality returns the desired concept, which is selected accordingly.
- Step 3: the user clicks on the “Integrate” button, which instantiates SPARQL Rule 1. Thus, the language integration is performed and stored in the triplestore. Figure 149 shows the query printed in the prompt command at the point in time the “integrate” button is clicked.

View 4 in Figure 148 shows the result of the extension, where the concept “Discretionary Task” is displayed as a sub-concept of “Manual Task”.

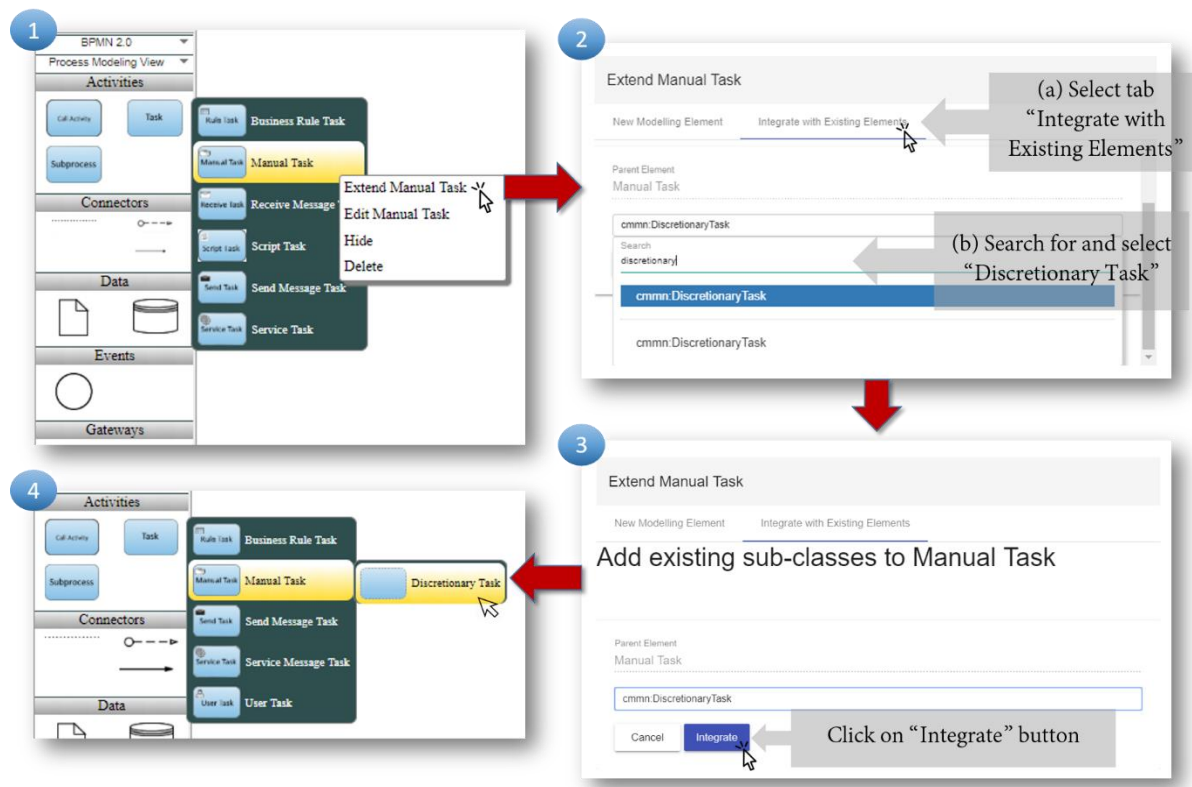


Figure 148. Steps to extend “Manual Task” from BPMN with “Discretionary Task” from CMMN

```
INSERT DATA {<http://ikm-group.ch/archimeo/cmnn#DiscretionaryTask> rdfs:subClassOf <http://ikm-group.ch/archimeo/BPMN#ManualTask> . <http://fhnw.ch/modelingEnvironment/PaletteOntology#DiscretionaryTask> po:hasParentPaletteConstruct <http://fhnw.ch/modelingEnvironment/PaletteOntology#ManualTask> . }***End query***
2019-08-06 19:07:41.57
```

Figure 149. Instance of SPARQL Rule 1 dynamically generated after adding Discretionary Task

7.4.1.4 Query Result After Implementing Use Case 1

In order to prove that the new concept has been entered successfully in to the ontology-based meta-model, Figure 150 shows the comparison of the query results both before and after the language integration. In particular, the upper part of both screenshots of Figure 150 contain a query asking for (1) the sub-class of Manual Task (i.e. *cmnn:DiscretionaryTask*), (2) and the

parent of the palette element *po:DiscretionaryTask* (see the query in Table 57). Retrieving this information proves that the SPARQL Rule 1 worked as expected. The lower parts of both screenshots in Figure 150 show the query results. In contrast to the result on the left-hand side of Figure 150, the one on the right-hand side contains the expected results: the class “*cmmn:DiscretionaryTask*” as a sub-class of “*bpmn:ManualTask*”, as well as the related palette element “*po:ManualTask*” being parent of the palette element “*po:DiscretionaryTask*”. The class extension is, therefore, correctly propagated to the ontology.

The graphical representation of the modelling language (see Figure 148) is consistent with the knowledge contained in the triplestore. Hence, for the considered functionality, the human- and machine interpretable representations of the modelling language have proven to be consistent one another. In turn, the functionality “Extending Modelling Constructs via Integration” is validated.



Figure 150. Query results (a) before and (b) after domain-specific adaptations in Use Case 1: Adding Discretionary Task

Table 57. SPARQL query to prove consistency in Use Case 1: Adding Discretionary Task

```
PREFIX po: <http://fhnw.ch/modellingEnvironment/PaletteOntology#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX bpmn: <http://ikm-group.ch/archiMEO/BPMN#>
SELECT DISTINCT ?class ?paletteElement ?parentPaletteElement
WHERE {
  ?class rdfs:subClassOf bpmn:ManualTask .
  ?paletteElement po:paletteConstructIsRelatedToModellingLanguageConstruct ?class .
  ?paletteElement po:paletteConstructHasParentPaletteConstruct ?parentPaletteElement .
}
```

7.4.2 Validation of Functionalities for Creating New Modelling Constructs, Domain Ontology Concepts and Semantic Mappings

This sub-section describes the validation of the AOAME’s functionalities that allow the user to extend a modelling language. Specifically, the functionalities of interest are: “Extending Modelling Construct”, “Creating New Domain Ontology concepts” and “Creating New Semantic Mappings”. This use case is extracted from the Business Process as a Service (BPaaS) case (see Section 4.2), and shows how a new modelling element is created as an extension of an existing one. This use case is also published (Laurenzi, Hinkelmann, & van der Merwe, 2018).

7.4.2.1 Description of Use Case 2: Extending BPMN Group

In this use case, the cloud broker uses BPMN to model a “Send Invoice” business process as well as annotating the process with business functional requirements (BPRs). In this use case, BPRs are pre-defined directly in the meta-model of BPMN. The reason for this is to both facilitate and speed-up the annotations of the business process. Figure 151 depicts an example of the pre-defined BPRs annotations over the Send Invoice business process. Just as in the BPaaS case, such annotation is performed to retrieve the most suitable cloud services for the given requirements.

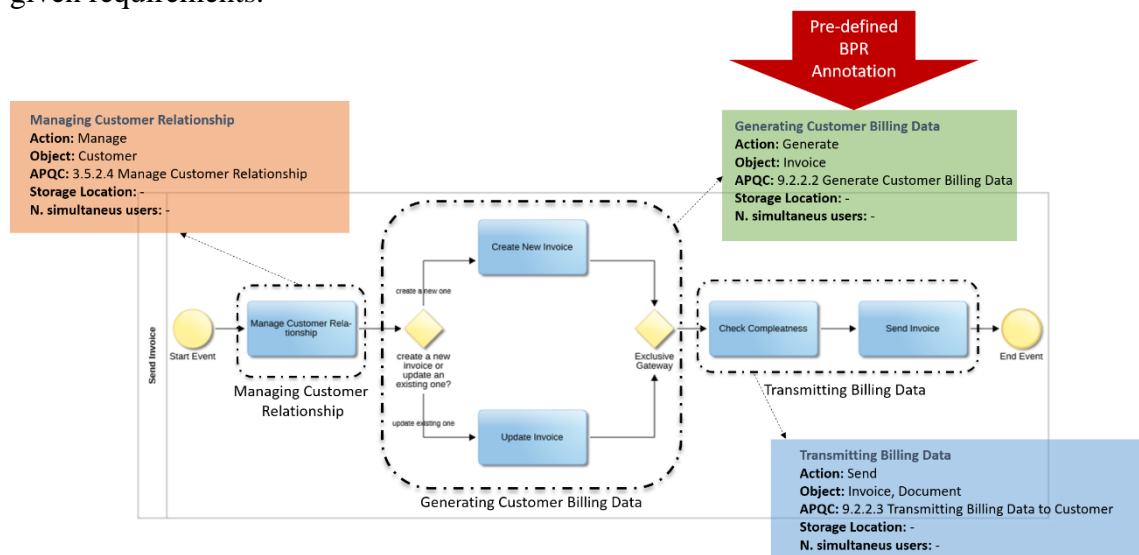


Figure 151. Process annotation with pre-defined requirements

As shown in Figure 151, the solution foresees the possibility for the cloud broker to use pre-defined specifications of business process requirements to annotate activities such as:

1. Customer relationship management,
2. Creation and update invoices (see red arrow in Figure 151),
3. Check for completeness and send invoices.

To support that, the BPMN modelling construct “Group” is extended to be further specified with domain-specific aspects such as pre-defined values of APQC categories, a verb (i.e. action) and a noun (i.e. object). Figure 152 shows the aspects of the extended modelling elements:

- “Managing Customer Relationship”: this modelling element is suitable to annotate BPMN tasks concerning the customer relationship management. It should be mapped to (1) the APQC category “3.5.2.4 Manage Customer Relationship”, (2) Action: “Manage”, and (3) Object: “Customer”.
- “Generating Customer Billing Data”: this modelling element is suitable to annotate BPMN tasks concerning the creation and/or update of invoices. It should be mapped to (1) the APQC category “9.2.2.2 Generate Customer Billing Data”, (2) Action: “Generate”, and (3) Object: “Invoice”.
- “Transmitting Billing Data”: this modelling element is suitable to annotate BPMN tasks concerning the transmission of invoices. It should be mapped to (1) the APQC category “9.2.2.3 Transmitting Billing Data”, (2) Action: “Send”, and (3) Object: “Invoice” or “Document”.

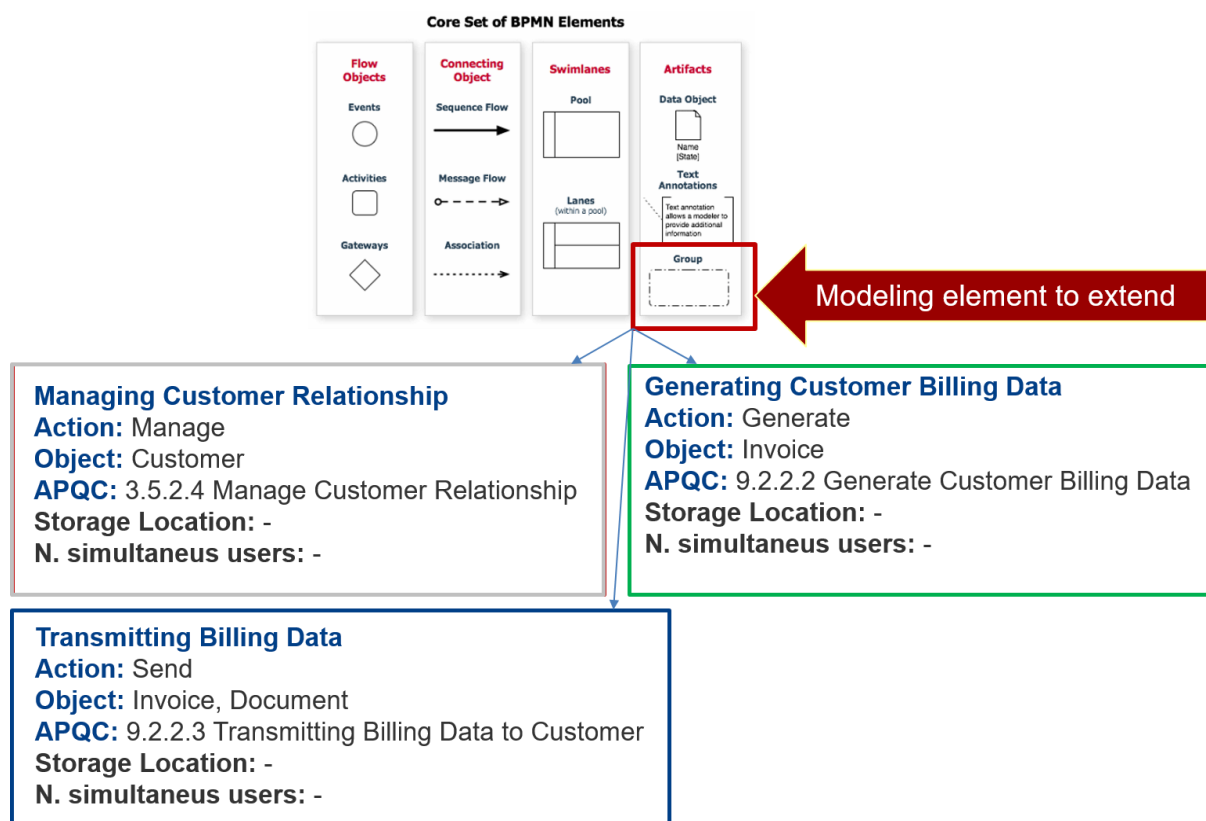


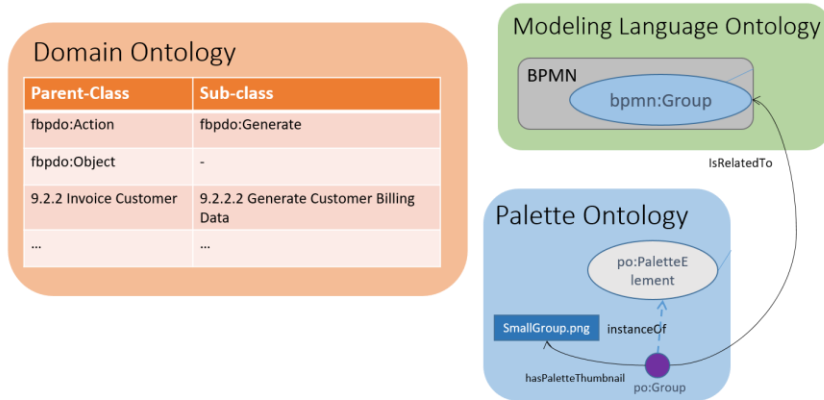
Figure 152. Extension of BPMN modelling element "Group" with three (domain-specific) sub-concepts

The conceptual solution of such an extension is depicted in Figure 153: the upper part of the figure “(a)” shows the conceptual representation before the extension occurs; the bottom part of the figure “(b)” shows the new conceptual representation after the extension. Note that for a better readability Figure 153 shows only the modelling element “Generating Customer Billing Data” as an extension of “Group”. In particular, the modelling language extension is expected to generate the following knowledge:

- A new class *bpaas:GenerateCustomerBillingData* as a sub-class of *bpmn:Group* in the Modelling Language Ontology. A label and a comment for the new class should also be inserted.

- A new instance *po:GeneratingCustomerBillingData* with properties in the Palette Ontology. The properties are (1) a datatype property for the new graphical notation, (2) an object property *hasParent* pointing to the instance *po:Group*, and (3) an additional object property *isRelatedTo* connecting the new instance with the modelling element *bpaas:GenerateCustomerBillingData*.
- A new sub-concept for *fbpdo:Object* named *fbpdo:Invoice*. The latter resides in the Domain Ontology and is created to be mapped with the new class *bpaas:GenerateCustomerBillingData*.
- New object properties for the semantic mappings between the new class *bpaas:GenerateCustomerBillingData* and the above suggested APQC category, object and action. The APQC categories, objects and actions contribute to specify the domain of interest, therefore are considered as concepts in the Domain Ontology.

(a) Before domain-specific adaptations



(b) After domain-specific adaptations

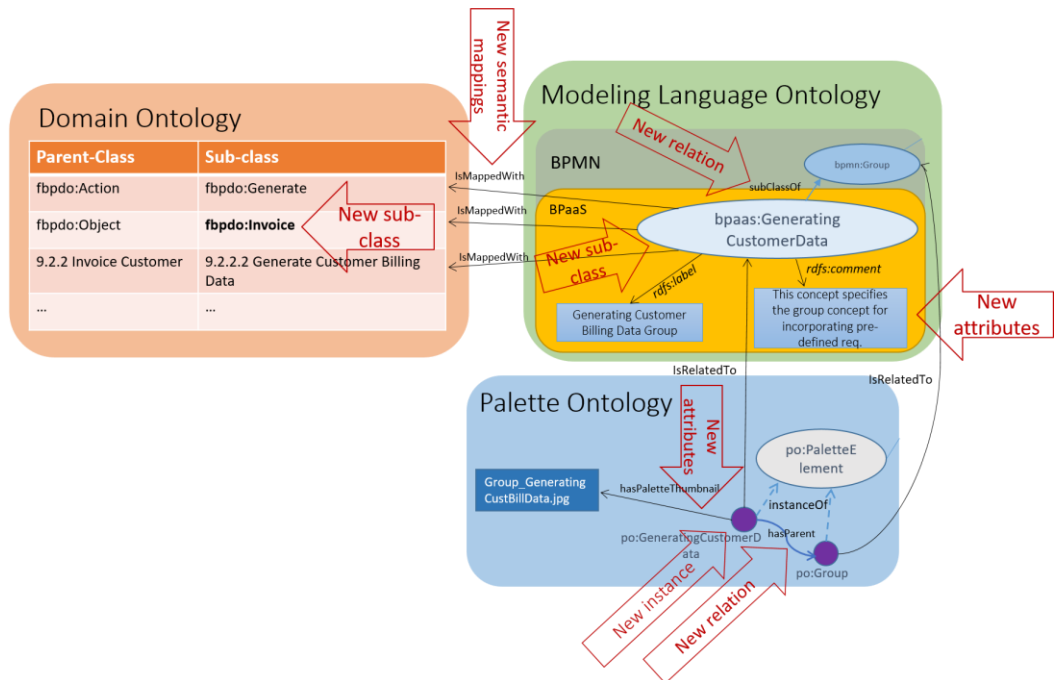


Figure 153. Conceptual solution (a) before and (b) after domain-specific adaptations for Extending BPMN Group use case

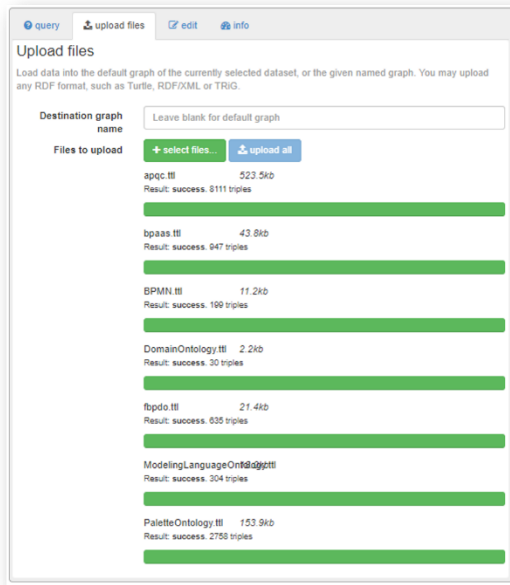
7.4.2.2 Set of Ontologies for Use Case 2

The set of ontologies required to implement Use Case 2 are as follows:

- The BPMN Ontology (see Use Case 1).
- The BPaaS Ontology, which contains the class hierarchy, attributes and relations of BPaaS (see Section 4.2). Since BPaaS extends BPMN, the file BPMN is imported in the BPaaS Ontology;
- The Modelling Language Ontology (see Sub-section 5.3.2.1). Both the BPMN Ontology and the BPaaS Ontology are imported and integrated in the Modelling Language Ontology. Modelling elements from both ontology files are entered as sub-classes of *lo:ModellingElement*, whilst modelling relations are entered as sub-classes of *lo:ModellingRelation*. The modelling language BPaaS has several modelling views. The excerpt shown on the top-right green corner of Figure 146 applies in this case too as it includes the BPMN Ontology.
- The APQC Ontology, which contains the ontology reflecting the class hierarchy of the APQC Process Classification Framework (APQC 2014);
- The FBPDO Ontology (**F**unctional **B**usiness **P**rocess **D**escription **O**ntology), which contains the class hierarchy of objects and actions. The combination of objects and actions is used to describe functional requirements and specifications of process activities or groups of activities (see Sub-section 4.2.6.4.4);
- The Domain Ontology, which is described in Sub-section 5.3.2.1. Both APQC Ontology and FBPDO Ontology are imported in the Domain Ontology.
- The Palette Ontology, which is described in Sub-section 5.3.2.1. In this case, the two classes *po:PaletteConnector* and *po:PaletteElement* contain the instances for displaying the graphical notations of BPMN. Just as in Use Case 1, the instances also contain the relations to the respective classes in the Modelling Language Ontology.

Identical to the previous use case, the above set of ontologies is uploaded to the triplestore in order to populate the palette (see left-hand side of Figure 154). The subsequent selections of the BPaaS DSML and the Process Modelling View enable the population of the palette (see the right-hand side of Figure 154). Note that the displayed graphical notations in the palette are the same as for BPMN 2.0. This is correct since BPaaS extends BPMN.

(a) Upload the set of ontologies on the triplestore



(b) Select modeling language/view to populate the Palette Component

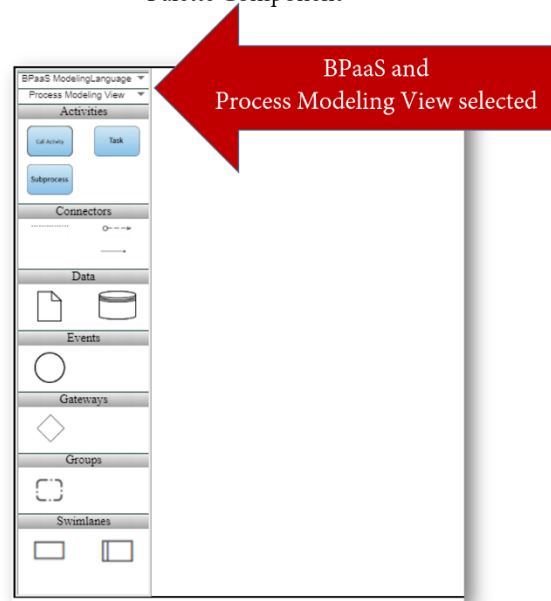


Figure 154. Populating the palette with graphical notations from the “Process Modelling View” of BPaaS

7.4.2.3 User actions performed in the Graphical User Interface

The description of the user actions is underpinned by Figure 155. The figure shows nine steps through which the modelling extension is performed. As afore-mentioned, the desired conceptual result is shown on the bottom of Figure 155. The nine steps are detailed below.

- Step 1: the user right-clicks on the graphical notation “Group” in the palette and then click on “Extend Group”. This opens the pop-up window for the element extension.
- Step 2: the following specifications are entered in the pop-up window: (1) the prefix for the new element “bpaaS”; (2) the name of the new element “Generating Customer Billing Data”; (3) a comment for the new element “This concept specifies ...”; (4) the graphical notation to display in the palette; and (5) the graphical notation to display in the Model Editor. Thus, the “Create New Modelling Element” button is clicked to create the new modelling element with the mentioned specifications. The view in Step 3 opens subsequently.
- Step 3: from this view, the “Semantic Mapping” tab is selected. This opens the view that allows adding the semantic mappings. Once the “Insert new Semantic Mapping” button is clicked, the view in step 4 opens.
- Step 4: in this step the new concept “Invoice” should be created as it does not exist yet in the Domain Ontology. Hence, the user clicks on the “Create New Domain Element” button, which leads to the view in step 5.
- Step 5: the new concept *fbpdo:Invoice* has to be entered as a sub-class of “Object”. Thus, the class “Object” is typed in the search box. Once the concept is returned, it is selected.

- Step 6: the name “Invoice” is entered for the new sub-class. Subsequently, the user clicks on the “Ok” button to save the new concept and proceeds to Step 7 (note that step 7 and step 4 show the same view but in different point in time).
- Step 7: the new domain element “Invoice” is searched and selected. The user can, therefore, enter the name of the new object property “isMappedWithInvoice”. In the ontology, this object property will have *bpaas:GeneratingCustomerBillingData* as a domain and *fbpdo:Invoice* as a range. The “Create Relation” button is then clicked so to store the new object property in the triplestore. This action leads to the view shown in Step 8. The dynamically generated SPARQL categorises this object property as a sub-property of *lo:elementIsMappedWithDOConcept*.
- Step 8: the view in this step is the same as the one in Step 3. The difference is that the saved object property (“isMappedWithInvoice”) is now displayed.
- Step 9: the graphical notation of the new Billing element is displayed as a sub-concept of “Group” in the palette. The new element appears as soon as the cursor is moved on the graphical notation.

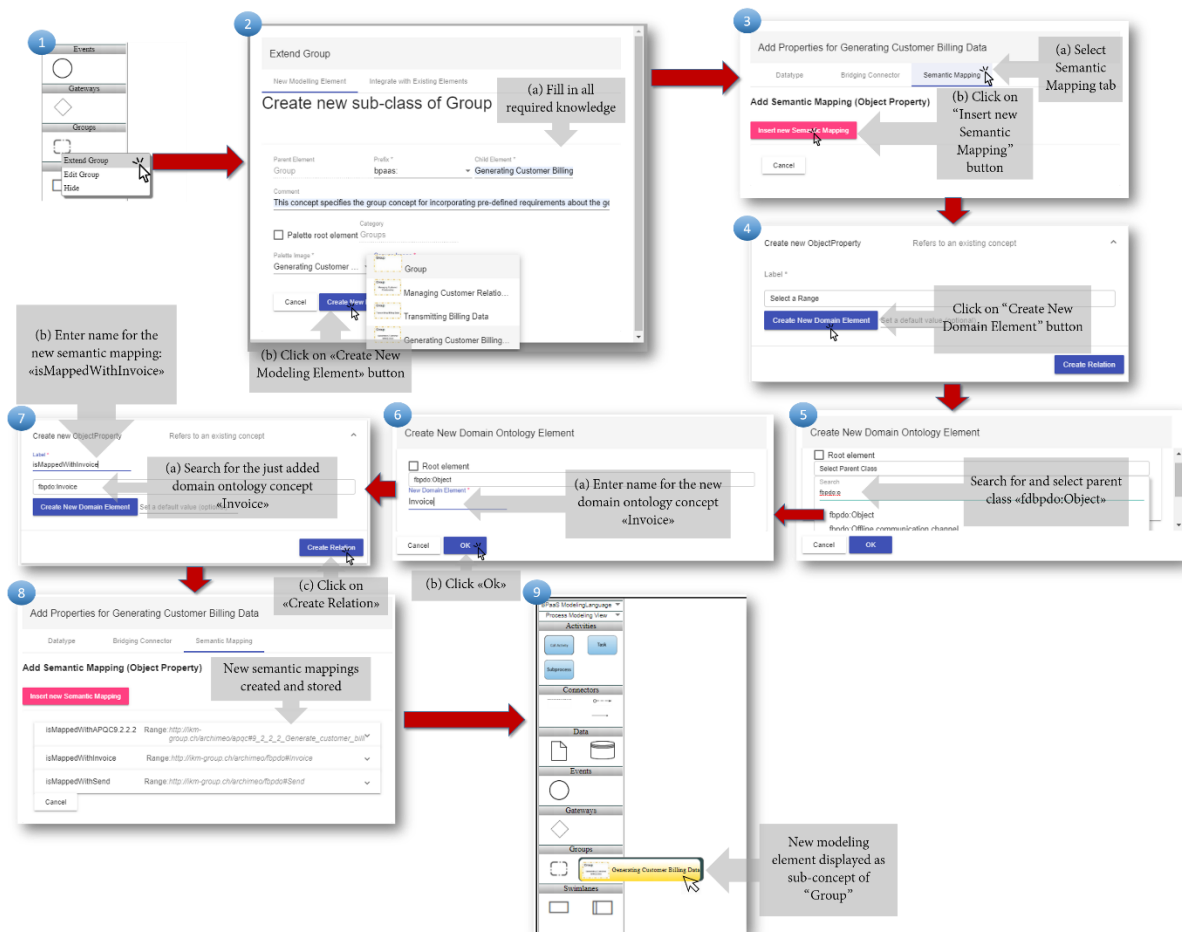


Figure 155. Steps to extend the BPMN “Group” with BPR annotation “Generating Customer Billing Data”

7.4.2.4 Query Result After Implementing Use Case 2

The three SPARQL rules that are mentioned at the beginning of the use case ensure the knowledge propagation to the triplestore. In order to prove that the ontology matches with the expected conceptual solution, the SPARQL query shown in Table 58 is fired against the ontologies. The query result is shown at the bottom of Figure 156. Due to the many specifications, only the result of the query performed after the language adaptation is reported. The complete screenshot can be found in Appendix F: Evaluation, folder F1.

From left to right, the query result in

shows the following specifications:

- The label of the newly created palette element: “Generating Customer Billing Data”. Rows 8 and 9 in Table 58 allow retrieval of this specification.
- The file name of the graphical notation for the new palette element. The ontology contains also the file name of the graphical notation for the Model Editor. It is, however, left out for a better readability of the query result. Row 10 in Table 58 allows retrieval of this specification.
- The comment of the new palette element. Row 11 in Table 58 allows retrieval of this specification.
- The label of the parent instance of the newly created palette element, i.e. “Group”. Rows 12 and 13 of Table 58 allow retrieval of this specification.
- The category to which the new palette element belongs. Row 14 in Table 58 allows retrieval of this specification.
- The default values “false” for the datatype property *po:paletteConstructIsHiddenFromPalette*, which is assigned to the created palette element. Row 15 in Table 58 allows retrieval of this specification.
- The created class *bpaas:GeneratingCusotmerBillingData* to which the new palette element is related. Rows 16 in Table 58 allows retrieval of specification.
- The newly added domain element *fbpdo:Invoice*. Row 17 and 18 in Table 58 allow retrieval of this specification.
- The parent class *bpmn:Group* of the newly added class *bpaas:GeneratingCusotmerBillingData* (note the different prefix in the two classes (bpaas and bpmn), which distinguishes the two ontologies). Row 19 in Table 58 allows retrieval of this specification.

The retrieved knowledge demonstrates the correct propagation of the language extension into the triplestore. Such extension included both (1) the creation and association of annotation properties and object property (i.e. semantic mapping), and (2) the creation and association of domain concepts. The graphical representation of the modelling language, which results from the user actions (see Figure 155), is consistent with the generated machine-interpretable knowledge. Therefore, for the considered functionalities, consistency between the human- and machine interpretable representation of the modelling language is proved. In turn, the functionalities “Creating New Modelling Constructs”, “Creating New Domain Ontology Concepts” and “Creating New Semantic Mappings” are validated.

```

6
7 SELECT DISTINCT ?paletteElementLabel ?graphicalNotation4Palette ?comment ?parentPaletteElementLabel ?paletteCategory ?isHidden ?
relatedClass ?newDomainConcept ?superClassOfNewModelingElement
8
9 WHERE {
10 ?paletteElement rdf:type po:PaletteElement .
11 bpaas:GeneratingCustomerBillingData rdfs:comment ?comment .
12 ?paletteElement po:paletteConstructHasPaletteThumbnail ?graphicalNotation4Palette .
13 ?paletteElement rdfs:label ?paletteElementLabel .
14 ?paletteElement po:paletteConstructHasParentPaletteConstruct ?parentPaletteElement .
15 ?parentPaletteElement rdfs:label ?parentPaletteElementLabel .
16 ?paletteElement po:paletteConstructIsGroupedInPaletteCategory ?paletteCategory .
17 ?paletteElement po:paletteConstructIsHiddenFromPalette ?isHidden .
18 ?paletteElement po:paletteConstructIsRelatedToModelingLanguageConstruct ?relatedClass .
19 ?relatedClass rdfs:label ?modelingElementLabel .
20 FILTER (?modelingElementLabel = "Generating Customer Billing Data") .

```

Expected result met

QUERY RESULTS

Showing 1 to 1 of 1 entries Search: Show 1000 entries

paletteElementLe	graphicalNotation	comment	parentPaletteEle	paletteCategory	isHidden	relatedClass	newDomainConc	superClassOfNew
1 "Generating Customer Billing Data"	"Group_GeneratingCustBillData.jp	"This concept specifies the group concept for incorporating pre-defined requirements about the generation of customer billing data"	"Group"	po:Category_Groups4BPaaSProcessesModelingView	"false"	bpaas:GeneratingCustomerBillingData	fbpdo:Invoice	bpmn:Group

Showing 1 to 1 of 1 entries

Figure 156. Query results after domain-specific adaptations in Use Case 2: Extending BPMN Group

Table 58 SPARQL query to prove consistency in Use Case 2: Extending BPMN Group

```

PREFIX po: <http://fhnw.ch/modellingEnvironment/PaletteOntology#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX bpaas: <http://ikm-group.ch/archimeo/bpaas#>
PREFIX lo: <http://fhnw.ch/modellingEnvironment/LanguageOntology#>
SELECT DISTINCT ?paletteElementLabel ?graphicalNotation4Palette ?comment
?parentPaletteElementLabel ?paletteCategory ?isHidden ?relatedClass ?newDomainConcept
?superClassOfNewModellingElement
WHERE {
?paletteElement rdf:type po:PaletteElement .
?paletteElement rdfs:label ?paletteElementLabel .
?paletteElement po:paletteConstructHasPaletteThumbnail ?graphicalNotation4Palette .
bpaas:GeneratingCustomerBillingData rdfs:comment ?comment .
?paletteElement po:paletteConstructHasParentPaletteConstruct ?parentPaletteElement .
?parentPaletteElement rdfs:label ?parentPaletteElementLabel .
?paletteElement po:paletteConstructIsGroupedInPaletteCategory ?paletteCategory .
?paletteElement po:paletteConstructIsHiddenFromPalette ?isHidden .
?paletteElement po:paletteConstructIsRelatedToModellingLanguageConstruct ?relatedClass .
lo:isMappedWithInvoice rdfs:range ?newDomainConcept .
lo:isMappedWithInvoice rdfs:domain ?relatedClass .
bpaas:GeneratingCustomerBillingData rdfs:subClassOf ?superClassOfNewModellingElement .

?relatedClass rdfs:label ?modellingElementLabel .
FILTER (?modellingElementLabel = "Generating Customer Billing Data") .
}

```

7.4.3 Validation of Functionalities for Creating New Bridging Connectors and Datatype Properties

The AOAME's functionalities "Extend modelling construct", "Create semantic mapping", "Create bridging connector" and "Created datatype property" are used to implement the third use case. Use Case 3 is similar to the one introduced in Sub-section 5.3.4.2.9 and is extracted from the Patient Transferal Management case (Section 4).

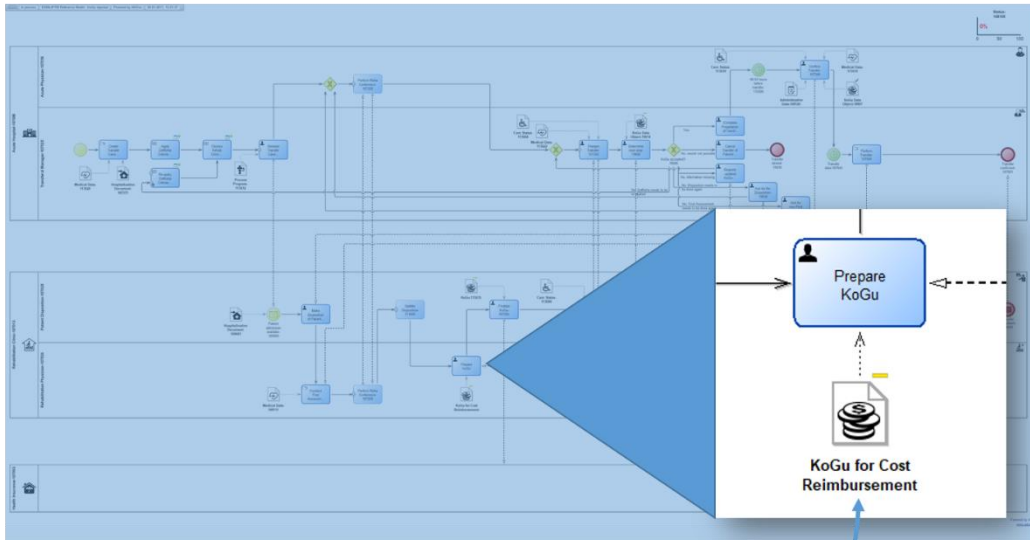
7.4.3.1 Description of Use Case 3: Adding ICF Standard Document

In this use case, the transferal manager needs to document medical information of patients according to the International Classification Of Functioning Disability Health (World Health Organisation, 2016), i.e. ICF Standard. Such medical information is required to be in the cost reimbursement request, which must be sent to the health insurance provider. A request that includes medical information of patients that does not comply with the standard is rejected. Therefore, the transferal manager needs a quick access to such medical information while preparing the request for the patient's case.

The *ICF Standard* is a specific document. Therefore, it is added as a specialisation of the concept *Data Document*. The latter belongs to the Document and Knowledge Modelling View of DSML4PTM (see bottom of Figure 157). On the other hand, the preparation of the cost reimbursement request is a BPMN User Task called *Prepare KoGu*. The request itself is modelled with a data object called *KoGu Data Object*, which is an input for the task *Prepare KoGu*. Both the user task and the data object belong to the Process Modelling View of DSML4PTM (see upper part of Figure 157). Next, given the need for the transferal manager to quickly access to medical information, a bridging connectore "is part of" is added between the *ICF Standard* document and the *KoGu Data Object* (see Figure 157).

In this use case, we assume that both the ICF Standard and the connection to the *KoGu Data Object* are not yet available from the DSML4PTM. Thus, domain-specific adaptations are to be performed.

(a) Process Modeling View



(b) Document and Knowledge Modeling View

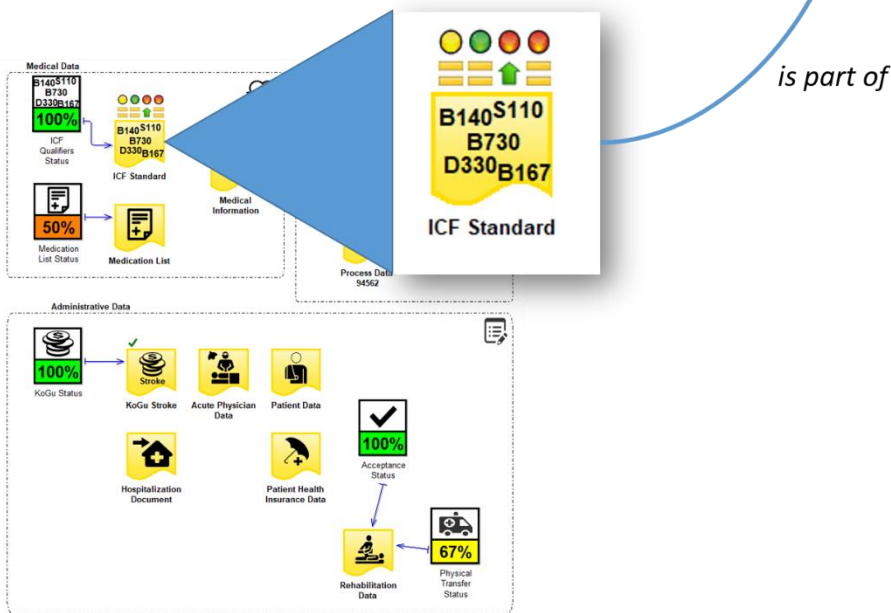
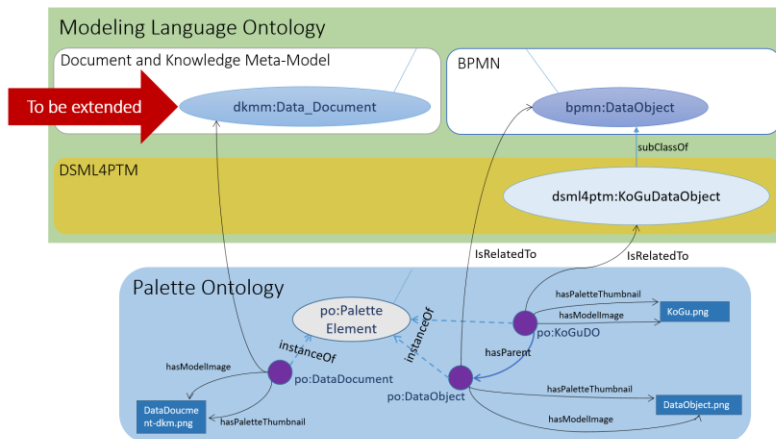


Figure 157. Bridging connector between the ICF Standard document and KoGu Data Object

Figure 158 depicts the conceptual solution of this use cases both (a) before and (b) after before and after the domain-specific adaptations, which are on top and bottom of the figure, respectively. The available elements and relations before the language adaptation are: *dkmm:Data_Document* from the Document and Knowledge Meta-Model (DKMM); *dsml4ptm:KoGuDataObject* from DSML4PTM, and the parent class *bpmn:DataObject* from BPMN. The concept *dsml4ptm:KoGuDataObject* extends the “*bpmn:DataObject*”. Medical information regarding the ICF Standard is also available. They already exist in the ICF Ontology (National Center for Biomedical Ontology, 2012). Since this contains domain knowledge and is not a language per se, it is imported and integrated in the Domain Ontology.

Further properties could also be considered in the ICF Standard, e.g. the patient progress status as well as general concepts like the physical location, which is defined in the Top Level Ontology (Emmenegger et al., 2013). However, they are left out to avoid stretching too much the use case.

(a) Before the domain-specific adaptations



(b) After the domain-specific adaptations

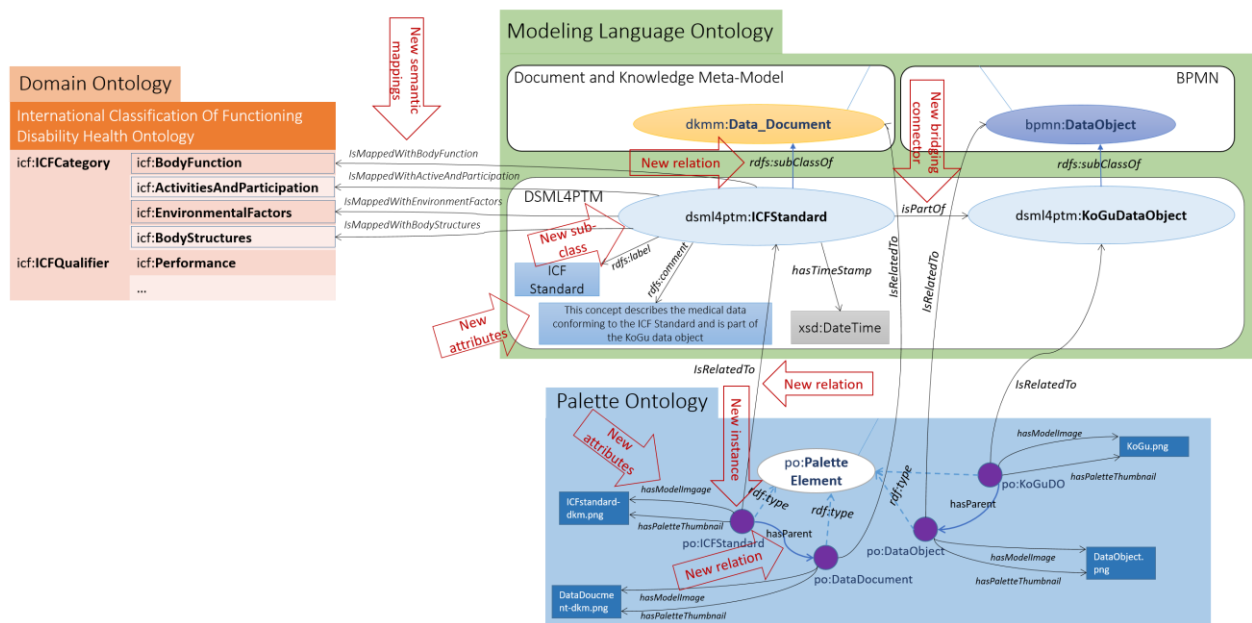


Figure 158. Conceptual solution (a) before and (b) after adding and referring ICF Standard document

Bottom of Figure 158 shows the expected result after performing the domain-specific adaptations. The result contains the following new resources:

- a new class *dsm14ptm:ICFStandard* as a sub-class of *dkmm:Data_Document*.
- four semantic mappings having *dsm14ptm:ICFstandard* as a domain and Body Function, Activities and Participation, Environmental Factors, Body Structures as ranges.
- two properties for the new class *dsm14ptm:ICFStandard*, label and comment.

- one datatype property for specifying date and time in *dsml4ptm:ICFStandard*.
- an object property *dsml4ptm:icfStandardIsPartOfKoGuDataObject* (i.e. bridging connector), which has *dsml4ptm:ICFstandard* as a domain and *dsml4ptm:KoGuDataObject* as a range.
- one instance *po:ICFstandard* of the class *po:PaletteElement*.
- one object property to denote that the instance *po:ICFstandard* has the instance *po:Data_Document* as a parent.
- properties for the instance *po:ICFstandard* already filled with concrete values, which are relevant for displaying the graphical notation on the palette. Specifically, the name *ICF Standard Document*, file names for the two chosen graphical notations, the category inherited from the parent *po:Data_Document* (i.e. *po:Category_Document4DSML4PTMDocumentView*), the related class in the modelling language ontology *dsml4ptm:ICFStandard*, and the “false” value, which denotes that the element will be displayed on the palette.

7.4.3.2 Set of ontologies for Use Case 3

The set of ontologies required to implement Use Case 3 are as follows:

- The BPMN Ontology (see Use Case 1 and 2).
- The DKMM Ontology, which contains the class hierarchy, attributes and relations of the Document and Knowledge Meta-Model (DKMM).
- The DSML4PTM Ontology, which contains the class hierarchy, attributes and relations of the DSML4PTM. DSML4PTM extends, among others, BPMN and DKMM.
- The Modelling Language Ontology (see Sub-section 5.3.2.1). The three language ontologies BPMN and DKMM and DSML4PTM are imported and integrated in the Modelling Language Ontology. Modelling elements from the three ontologies are entered as sub-classes of *lo:ModellingElement* while modelling relations are entered as sub-classes of *lo:ModellingRelation*. The modelling language DSML4PTM has several modelling views, i.e. process modelling view, document and knowledge modelling view, organisation modelling view, Decision Modelling View and the control element modelling view. The green quadrant in Figure 159 shows an excerpt of the Modelling Language Ontology containing concepts of both DKMM and BPMN.
- An excerpt of the ICF Ontology, which contains a few concepts of the National Centre for Biomedical Ontology (2012).
- The Domain Ontology (Sub-section 5.3.2.1). The ICF ontology is imported in the Domain Ontology. The orange quadrant in Figure 159 shows an excerpt of the Domain Ontology containing concepts of the ICF Standard.
- The Palette Ontology (Sub-section 5.3.2.1). The two classes *po:PaletteConnector* and *po:PaletteElement* contains the instances for displaying the graphical notations of BPMN. Such instances are already linked to the respective classes in the Modelling Language Ontology. The blue quadrant in Figure 159 shows an excerpt of the Palette Ontology, which contains two palette elements *po:DataDocument* and *po:DataObjects*. As the figure shows, the two instances are linked with the homonym classes, which belong to the modelling languages DKMM and BPMN, respectively.

Just as in the previous use cases, this set of ontologies is uploaded to the triplestore in order to populate the palette. Differently from the previous two use cases, the selected modelling language for the palette is DSML4PTM (i.e. Domain Specific Modelling Language for Patient Transferal Management) and the selected modelling view is “Document and Knowledge Modelling View”.

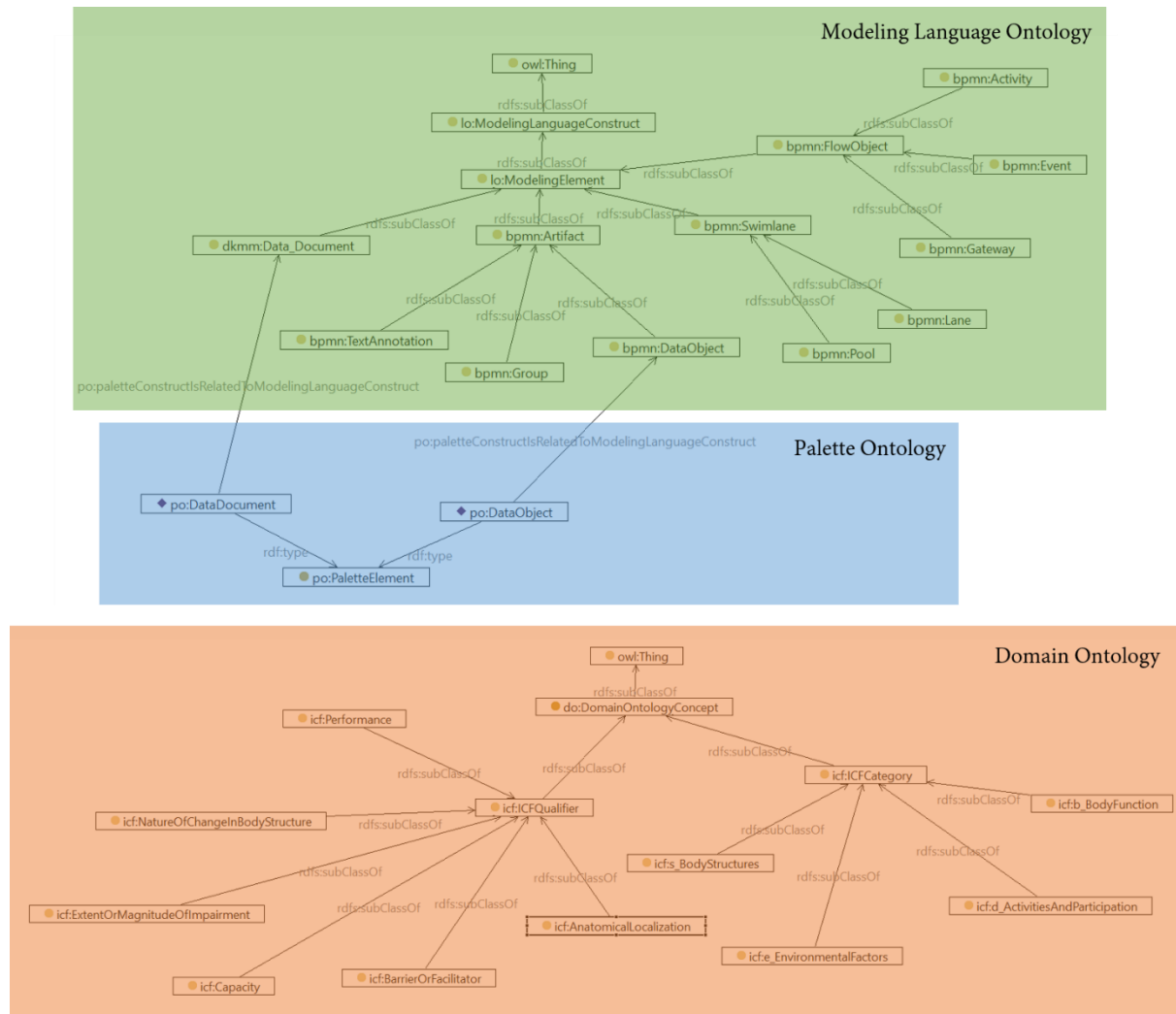


Figure 159. Excerpt of the Palette Ontology, Modelling Language Ontology and Domain Ontology related to the modelling languages DKMM, BPMN and ICF

7.4.3.3 User Actions Performed in the Graphical User Interface

Figure 160 depicts the user actions that are performed to implement Use Case 3. These are expressed in more detail below:

- Step 1: the user right-clicks on the graphical notation “Data Document” in the palette and then clicks on “Extend Group”. This action opens the pop-up window for the element extension.
- Step 2: the user enters the following specifications in the respective fields: (1) “dsml4ptm” as prefix, (2) “ICF International Standard” as a label for the new element, (3) “International Classification of Functioning, Disability and Health <https://bioportal.bioontology.org/ontologies/ICF?p=classes>” as a comment, (4) the graphical notation for the palette and (5) the graphical notation for the Model Editor. Then, once the “Create New Modelling Element” button is clicked, the new modelling element is created along with the entered specifications. In parallel, the Datatype Property view opens.
- Step 3: the user clicks on the “Insert new Datatype Property” button, which opens the view that allows new data type properties to be added.
- Step 4: the user enters the name for the new datatype property *ICFstandardHasTimeStamp* and the type of value for *DateTime*. Once the “Create Attribute” button is clicked, the property is created and stored in the triplestore.
- Step 5: the view shows the previously created Datatype property.
- Step 6: the user selects first the “Bridging Connector” tab and then clicks on the “Insert New Bridging Connector” button. This action opens the new view that allows the bridging connectors to be added.
- Step 7: The user types in the search box to look for the existing modelling element to add. The modelling element is retrieved from the ontology and subsequently selected. Next, the user enters “ICFStandardIsPartOfDataObject” as a name for the new object property. As soon as the button “Create Relation” is clicked on, the object property is created and stored in the triplestore. Moreover, the object property is entered as a sub-property of *lo:elementHasBridgingConnector*.
- Step 8: the view in this step shows the just entered bridging connector with its name and range.
- Step 9: similar to Use Case 2, the user selects the “Semantic Mapping” tab, and clicks on the “Insert new Semantic Mapping” button. The latter opens the view that allows new semantic mappings to be entered.
- Step 10: in this view the user types the desired domain concepts in the search box. Hence, one at a time, the selected domain elements are: Body Function, Body Structure, Environment Factors, and Activities and Participations. For each created property the user assigns a name, which is used to create the URI for the new property. The new object properties are entered as sub-properties of *lo:elementIsMappedWithDOConcept*.
- Step 11: all the four entered semantic mappings are presented in this view.
- Step 12: this view shows that the new modelling element has been added as sub-concept of Data Document and is displayed as soon as the cursor is moved on the graphical notation.

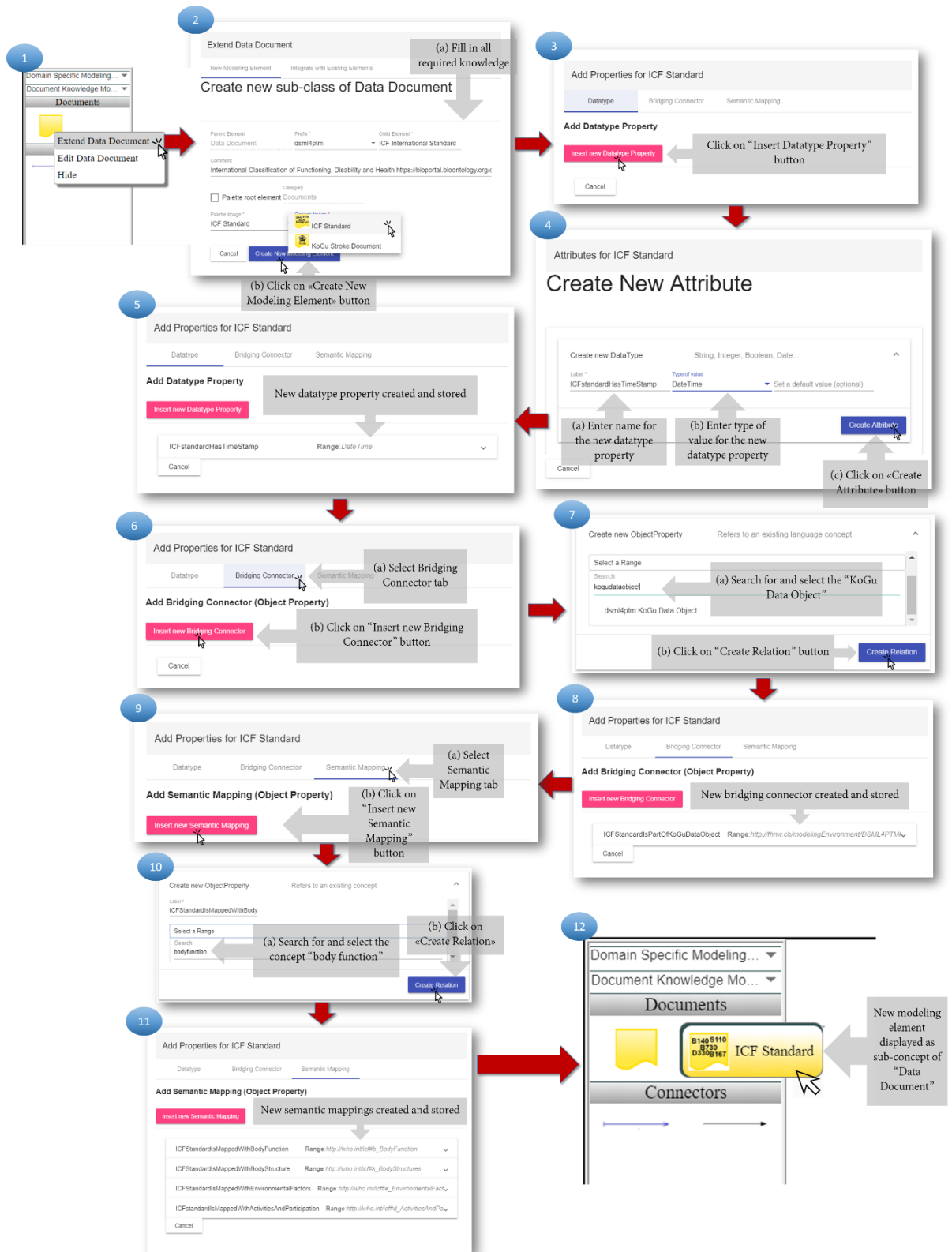


Figure 160. Steps to extend the modelling element “Data Document” with “ICF Standard”

7.4.3.4 Query Result After Implementing Use Case 3

The semantic rules (2, 3 and 4) that are incorporated in the functionalities ensure the propagation of the language adaptations to the ontology in the triplestore. The SPARQL query in Table 59 was executed to retrieve the newly produced knowledge from the triplestore. The query result is shown in the bottom part of Figure 161. Like for Use Case 2, due to the many specifications only the query result performed after the language adaptation is reported (the complete screenshot can be found in Appendix F, folder F1).

From left to right, the query result in Figure 161 shows the following specifications:

- The label of the newly created palette element: “ICF Standard”. Rows 9 and 10 of Table 59 allows retrieval of this specification.
- The file name of the graphical notation for the Palette. The ontology also contains the file name of the graphical notation for the Model Editor but it is not retrieved so as not to overcrowd the query result. Row 11 of Table 59 allows retrieval of this specification.
- The default values “false” for the datatype property *po:paletteConstructIsHiddenFromPalette*, which is assigned to the created palette element. Row 12 of Table 59 allows retrieval this specification.
- The category to which the new palette element belongs. Row 13 of Table 59 allows retrieval of this specification.
- The parent instance of the newly created palette element, i.e. “Data Document”. Row 14 of Table 59 allows retrieval of this specification.
- The created class *dsml4ptm:ICFStandard* to which the new palette element is related. Row 16 of Table 59 allows retrieval of this specification.
- The comment of the created class. Row 17 of Table 59 allows retrieval of this specification.
- The parent class *dkmm:Data_Document* of the newly added class *dsml4ptm:ICFStandard*. Note the different prefix of the two classes (dkmm and dsml4ptm), which distinguish the ontology files. Row 18 of Table 59 allows retrieval of this specification.
- The added datatype property *dsml4ptm:ICFStandardHasTimeStamp*. Rows 20 and 21 of Table 59 allow retrieving this specification.
- The chosen value type for the datatype property, i.e. “DateTime”. Rows 20 and 22 of Table 59 allow retrieval of this specification.
- The new object property added as a bridging connector: *dsml4ptm:ICFStandardIsPartOfKoGuDO*. Rows 24 and 25 of Table 59 allow retrieval of this specification.
- The modelling element *dsml4ptm:KoGuDataObject*, which is the range of the bridging connector. Rows 24 and 26 of Table 59 allow retrieval of this specification.
- One of the four object properties added as semantic mappings: *dsml4ptm:ICFStandardIsMappedWithBodyFunction*. Rows 28 and 29 of Table 59 allow retrieval of this specification. Note that the query result has four rows because of the four different semantic mappings, which are used to specify the modelling element ICF International Standard.
- The domain element *icf:b_BodyFunction*, which is the range of the above-mentioned semantic mapping. Rows 28 and 30 of Table 59 allow retrieval of this specification.

All the new above-listed ontology resources were created according to the domain-specific adaptations previously described. The new resources match with the desired outcome that was initially set (see Sub-section 7.4.3.1), thus the changes are propagated correctly. Moreover, the query result proves that the machine-interpretable representation of the modelling language is consistent with the graphical representation. The latter has been shown in the previous Sub-section (see Figure 160). Therefore, the remaining functionalities of Feature 1 (see Sub-section 6.3.1) “Creating New Bridging Connectors” and “Creating New Datatype Properties” are also validated.

6 SELECT DISTINCT ?paletteElementLabel ?graphicalNotation4Palette ?isHidden ?paletteCategory ?parentPaletteElement ?relatedClass
comment ?parentClassOfNewModelingElement ?attribute ?valueType ?bridgingConnector ?modelingElement ?semanticMapping ?DOconcept
7 WHERE {
8 ?paletteElement rdf:type po:PaletteElement .
9 ?paletteElement rdfs:label ?paletteElementLabel .
10 ?paletteElement po:paletteConstructHasPaletteThumbnail ?graphicalNotation4Palette .
11 ?paletteElement po:paletteConstructIsHiddenFromPalette ?isHidden .
12 ?paletteElement po:paletteConstructIsGroupedInPaletteCategory ?paletteCategory .
13 ?paletteElement po:paletteConstructHasParentPaletteConstruct ?parentPaletteElement .
14 ?paletteElement po:paletteConstructIsRelatedToModelingLanguageConstruct ?relatedClass .
15 ?relatedClass rdfs:comment ?comment .
16 ?relatedClass rdfs:subClassOf ?parentClassOfNewModelingElement .
17 ?attribute rdf:type owl:DatatypeProperty .
18 ?attribute rdfs:domain ?relatedClass.

QUERY RESULTS
Table Raw Response
Showing 1 to 4 of 4 entries Search: Show 1000 entries

paletteEler	graphicalIN	isHidden	paletteCati	parentPale	relatedClas	comment	parentClas	attribute	valueType	bridgingCc	modelingE	semanticM	DOconcept
1 "ICF International Standard"	"ICFStandard-dkm.png"	"false"	po:Category_DSM4PTMDocumentView	po:DataDocument_4DSML4PTM	<http://fnw.ch/modelingEnvironment/DSML4PTM#ICFStandard>	"International Classification of Functioning, Disability and Health https://biportal.biontology.org/ontologies/ICF?classes"	<http://fnw.ch/modelingEnvironment/dkmm#Data_Document>	<http://fnw.ch/modelingEnvironment/DSML4PTM#ICFStandardHasTimeStamp>	<http://www.w3.org/2001/XMLSchema#dateTime>	<http://fnw.ch/modelingEnvironment/DSML4PTM#ICFStandardIsPartOfKoGuDataObject>	<http://fnw.ch/modelingEnvironment/DSML4PTM#ICFStandardWithBodyFunction>	<http://who.int/icfbodyfunction>	

Expected result met

Figure 161. Query results after domain-specific adaptations in Use Case 3: Adding and Referring ICF Standard document

Table 59 SPARQL query to prove consistency in Use Case 3: Adding and Referring ICF Standard Document”

```

PREFIX lo: <http://fhnw.ch/modellingEnvironment/LanguageOntology#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX po: <http://fhnw.ch/modellingEnvironment/PaletteOntology#>
SELECT DISTINCT ?paletteElementLabel ?graphicalNotation4Palette ?isHidden ?paletteCategory
?parentPaletteElement ?relatedClass ?comment ?parentClassOfNewModellingElement ?attribute
?valueType ?bridgingConnector ?modellingElement ?semanticMapping ?DOconcept

WHERE {
?paletteElement rdf:type po:PaletteElement .
?paletteElement rdfs:label ?paletteElementLabel .
?paletteElement po:paletteConstructHasPaletteThumbnail ?graphicalNotation4Palette .
?paletteElement po:paletteConstructIsHiddenFromPalette ?isHidden .
?paletteElement po:paletteConstructIsGroupedInPaletteCategory ?paletteCategory .
?paletteElement po:paletteConstructHasParentPaletteConstruct ?parentPaletteElement .

?paletteElement po:paletteConstructIsRelatedToModellingLanguageConstruct ?relatedClass .
?relatedClass rdfs:comment ?comment .
?relatedClass rdfs:subClassOf ?parentClassOfNewModellingElement .

?attribute rdf:type owl:DatatypeProperty .
?attribute rdfs:domain ?relatedClass.
?attribute rdfs:range ?valueType.

?bridgingConnector rdfs:subPropertyOf lo:elementHasBridgingConcept .
?bridgingConnector rdfs:domain ?relatedClass.
?bridgingConnector rdfs:range ?modellingElement.

?semanticMapping rdfs:subPropertyOf lo:elementIsMappedWithDOConcept .
?semanticMapping rdfs:domain ?relatedClass.
?semanticMapping rdfs:range ?DOconcept .

?relatedClass rdfs:label ?relatedClassLabel .
FILTER (?modellingElementLabel = "ICF International Standard") .
} ORDER BY ?DOconcept

```

7.4.4 Validation of Functionalities for Deleting Modelling Constructs and Properties

This sub-section describes the validation of the two AOAME's delete functionalities "Deleting Modelling Construct" and "Removing Datatype Properties, Bridging Connectors, Semantic Mappings". For this purpose, a use case is extracted from the Patient Transferal Management application domain. This use case is a follow-up of the previously described Use Case 3 (see Sub-section 7.4.3). The set of ontologies for Use Case 3 (see Sub-section 7.4.3.2) are identical for this use case. Their description is, therefore, omitted in this sub-section.

7.4.4.1 Description of Use Case 4: Deleting ICF Standard document and/or properties

This use case is split into two sub-use cases, where in the first, the transferal manager does not need the ICF Standard document any longer, and thus wants it to be deleted. In the second sub-use case the transferal manager keeps the ICF Standard document but wants to remove the bridging connector with the KoGu Data Object.

Figure 162 shows the conceptual representation of the use case, where:

1. the arrow named "(1) To delete" indicates all the resources that should be deleted as soon the modelling element "ICF Standard" is deleted. As Figure 162 shows, such adaptation updates both the Modelling Language Ontology and the Palette Ontology.
2. the arrow named "(2) To delete" indicates the resource to be deleted once the bridging connector "isPartOf" is deleted. In this case, only the Modelling Language Ontology is affected.

The deletion of the modelling element is described below, while the deletion of the bridging connector is described in Sub-section 7.4.4.1.3.

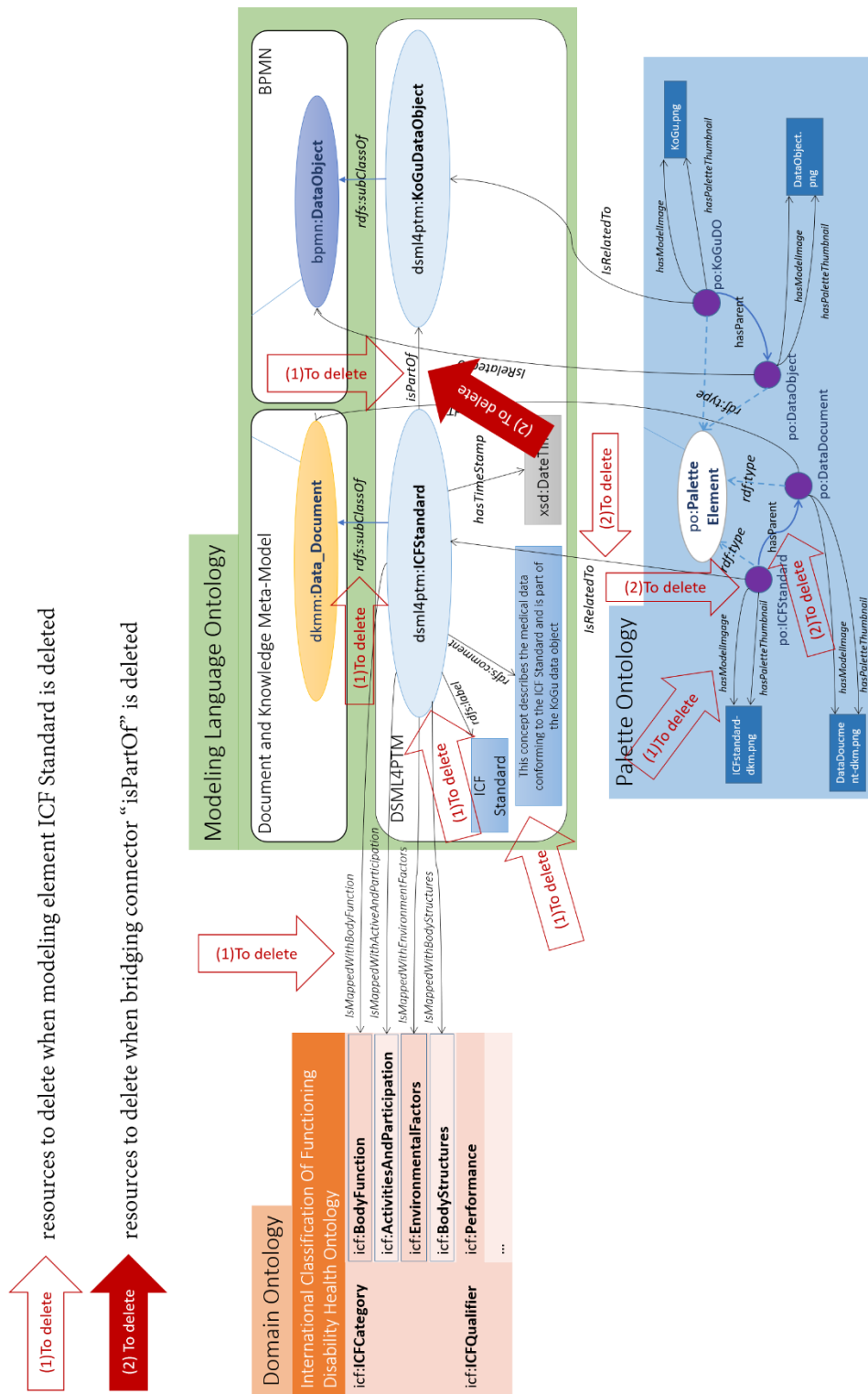


Figure 162. Conceptual solution for Use Case 4: Deleting ICF Standard document and properties

7.4.4.1.1 Deleting ICF Standard document

In the following, the three steps illustrate the user actions taken to delete the modelling element ICF Standard. Figure 163 graphically depicts the three steps.

- Step 1: the user right-clicks on the modelling element ICF Standard and subsequently clicks on “Delete”. Then, the window pop-up asking for confirmation of the removal appears.
- Step 2: the user clicks on the “OK” button in order to confirm the removal of the selected modelling element.
- Step 3: the modelling element is deleted from the triplestore. In result, the graphical notation is no longer displayed in the palette.

Note that the modelling element can be removed because it does not have any sub-concepts. As already stated in Sub-section 0, all modelling elements that have sub-concepts are not allowed to be deleted to avoid unintentionally deleting classes that may be wanted.

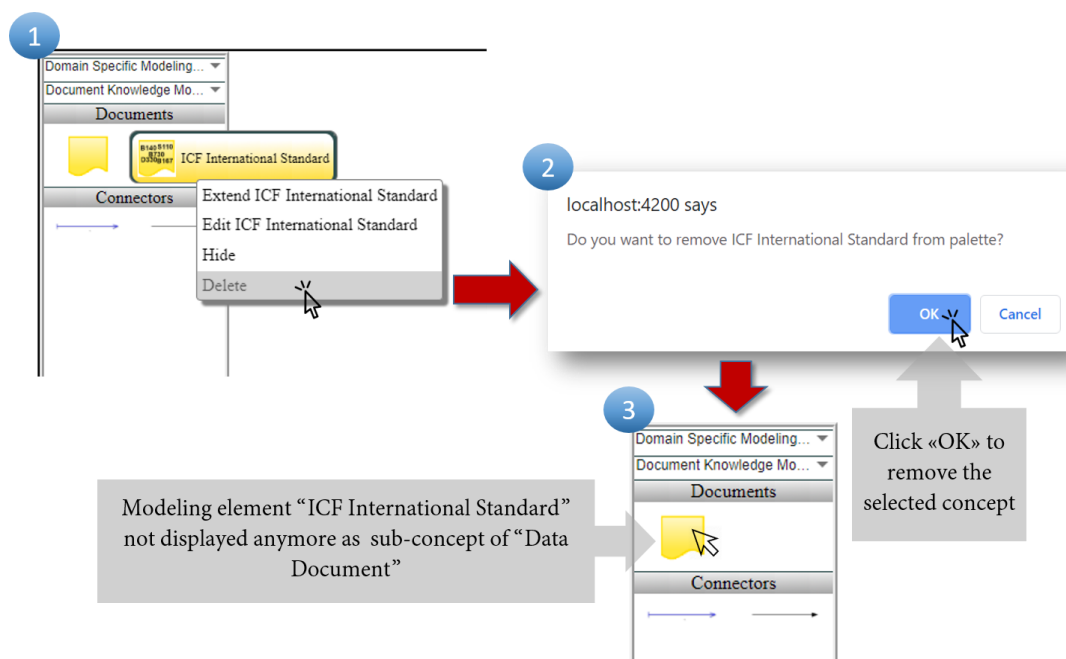


Figure 163. Steps to delete the modelling element ICF Standard

7.4.4.1.2 Query Result After Implementing Use Case 4 - Deleting a Modelling Construct

The dynamic instantiation of SPARQL Rule 6, ensures that the user actions are correctly propagated in the ontology. To prove consistency between the graphical modelling language and the ontology, the query in Table 60 was fired both before and after deleting the modelling element. In particular, the SPARQL query asks for (see also Table 60):

- all properties of the modelling element *dsml4ptm:ICFStandard* (see *?predicateOfME*) and all objects to which these properties are associated (see *?objectOfME*);
- all properties of the palette element *po:ICFStandard* (see *?predicateOfPE*) and all objects to which these properties are associated (see *?objectOfPE*).

Both query results are depicted in Figure 164. The query result on the left-hand side of Figure 164 shows an excerpt of the 40 retrieved entries. For instance, the first entry contains the following:

- In the first two columns there are the property *rdfs:subClassOf* and the class *dkmm:Data_Document*, which states that the class *dsml4ptm:ICFStandard* is sub-class of *dkmm:Data_Document*.
- In the third and fourth columns there are the property *po:paletteConstructIsRelatedToModellingLanguageConstruct* and the class *dsml4ptm:ICFStandard*, which state that the palette element *po:ICFStandard* (from the Palette Ontology) is related to the class *dsml4ptm:ICFStandard* (from the Modelling Language Ontology).

The complete list of the retrieved entries can be found in Appendix F: Evaluation, folder 3.

In contrast, the query result on the right-hand side of Figure 164 is empty. Note that among the deleted resources there is the property *rdf:type* for both *dsml4ptm:ICFStandard* and *po:ICFStandard*:

- *rdf:type* for *dsml4ptm:ICFStandard* states that the latter is a class;
- *rdf:type* for *po:ICFStandard* states that the latter is an instance of the class *po:PaletteElement*.

These complete list of the 40 retrieved entries can be found in Appendix F: Evaluation, folder F2.

By deleting *rdf:type* in this case, both the class *dsml4ptm:ICFStandard* and the instance *po:ICFStandard* are deleted simultaneously. In order to prove that, an additional query was executed (see Table 61). The query retrieves the sub-class of Data Document as well as the instance of the class palette Element, which has the instance Data Document as a parent.

As shown in Figure 164, the query was performed before and after deleting the modelling element ICF Standard. The results of the new query are shown in Figure 165 and confirm that both the modelling element and the palette element are deleted. Also, the query result is consistent with the graphical representation of the language, which is depicted in Figure 163 of Sub-section 7.4.4.1. Hence, consistency between the human- and machine-interpretable knowledge is proved also for the functionality “Deleting Modelling Constructs”. The functionality is, therefore, validated.

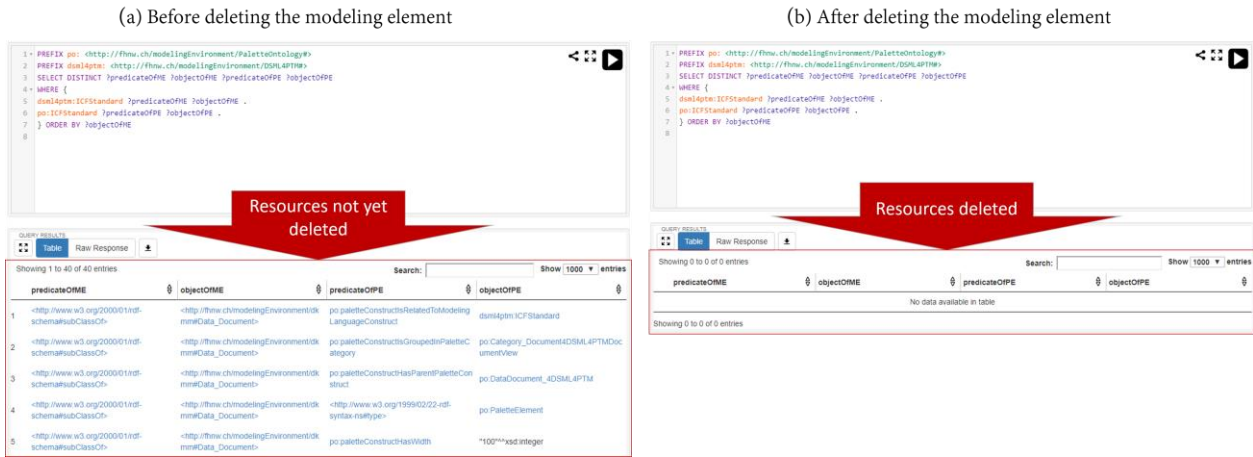


Figure 164. Query results about the properties (a) before and (b) after the deleting the ICF Standard document

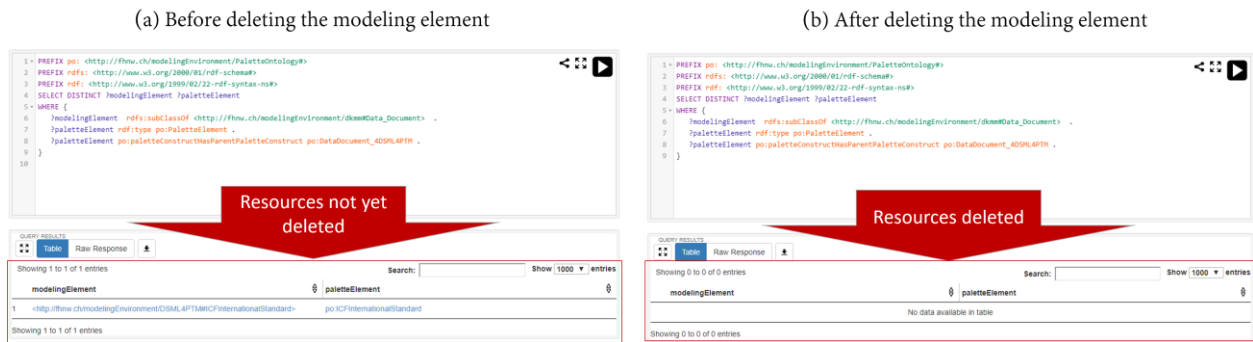


Figure 165. Query results about the modelling element (a) before and (b) after deleting ICF Standard document

Table 60 SPARQL query to prove that properties of ICF Standard document are deleted

```

PREFIX po: <http://fhnw.ch/modellingEnvironment/PaletteOntology#>
PREFIX dsml4ptm: <http://fhnw.ch/modellingEnvironment/DSML4PTM#>
SELECT DISTINCT ?predicateOfME ?objectOfME ?predicateOfPE ?objectOfPE
WHERE {
dsml4ptm:ICFStandard ?predicateOfME ?objectOfME .
po:ICFStandard ?predicateOfPE ?objectOfPE .
} ORDER BY ?objectOfME

```

Table 61. SPARQL query to prove that ICF Standard document is deleted

```

PREFIX po: <http://fhnw.ch/modellingEnvironment/PaletteOntology#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT DISTINCT ?modellingElement ?paletteElement
WHERE {
  ?modellingElement rdfs:subClassOf <http://fhnw.ch/modellingEnvironment/dkmm#Data_Document> .
  ?paletteElement rdf:type po:PaletteElement .
  ?paletteElement po:paletteConstructHasParentPaletteConstruct po:DataDocument_4DSML4PTM .
}

```

7.4.4.1.3 Deleting Bridging Connector “isPartOf”

The functionality “Removing Datatype Properties, Bridging Connectors and Semantic Mappings” allows the removal of datatype properties, bridging connectors and semantic mappings. The reference SPARQL Rule 7 is the same for all the three property types, independently from which property types is to be deleted. Therefore, showing the validation of one property type is enough as it would extend to the remaining two property types. The validation of the functionality, in this case, focuses on the bridging connector. Specifically, the functionality is used to delete the bridging connector “*isPartOf*”.

The steps are depicted in Figure 166 and are described as follows:

- Step 1: right-click on the modelling element of which property is to be deleted (i.e. ICF Standard) and click on “Edit ICF International Standard”. This leads to the pop-up window that allows editing or deleting properties of the selected modelling element.
- Step 2: select the “Bridging Connector” tab to show all the bridging connectors of the selected modelling element. As described in Sub-section 6.3.2.1, the bridging connectors are retrieved by a dynamically generated SPARQL SELECT.
- Step 3: click on the “Delete” button of the bridging connector that is to be deleted: *lo:ICFStandardIsPartOfKoGuDataObject*.
- Step 4: as expected, the bridging connector is no longer available as it has been removed.

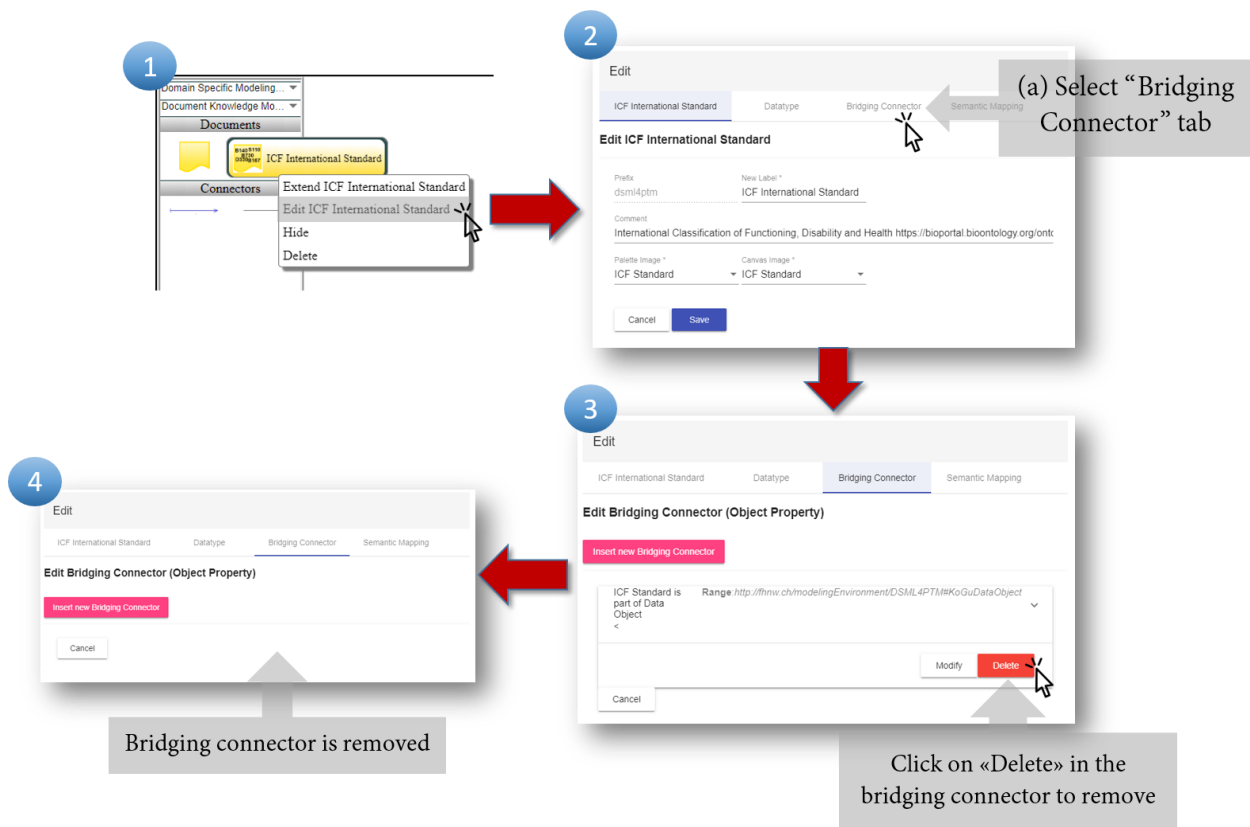


Figure 166. Steps required to delete a bridging connector (i.e. object property) in the GUI

7.4.4.1.4 Query Result After Implementing Use Case 4 - Deleting a Property

The dynamic instantiation of *SPARQL Rule 7*, ensures that the language adaptations are correctly propagated to the ontology. In this case also, in order to prove consistency the query shown in Table 62 was executed both before and after deleting the property. The SPARQL query asks for all properties associated with the bridging connector *lo:ICFStandardIsPartOfKoGuDO* (see *?predicate* in Table 62) and all objects (i.e. targeting resources) to which these properties are associated (see *?object* in Table 62).

Both query results are depicted in Figure 167. The query result on the left-hand side of Figure 167 shows the five retrieved entries. In particular:

1. The first entry contains the predicate *rdf:type* and the object *owl:ObjectProperty*, which specifies that the bridging connector is an object property.
2. The second entry contains the predicate *rdfs:label* and the object "ICF Standard is part of Data Object", which specifies the label of the bridging connector.
3. The third entry contains the predicate *rdfs:range* and the object *dsml4ptm:KoGuDataObject*, which specifies the range (i.e. target) of the bridging connector.
4. The fourth entry contains the predicate *rdfs:domain* and the object *dsml4ptm:ICFStandard*, which specifies the domain (i.e. source) of the bridging connector.

- The fifth entry contains the predicate *rdfs:subPropertyOf* and the object *lo:elementHasBridgingConcept*, which specifies that the bridging connector is a sub-property of the mentioned object property.

The right-hand side of Figure 167 shows the result of the query being executed after the deletion. The query returns an empty result, which proves that the bridging connector is removed. The result is consistent to the graphical representation that has been shown in Figure 166. Therefore, the consistency between the graphical representation of the modelling language and its machine-interpretable representation is proven for the removal of bridging connectors.

The same above-seen steps are applied to also remove both datatype properties and semantic mappings. The difference is only in the content of the steps from 2 to 4. For instance, removing a datatype property implies the selection of the “Datatype” tab in *step 2*. The same is for removing a semantic mapping, which requires the selection of the “Semantic Mapping” tab.

As already mentioned, the functionality instantiates the same rule for each property type, thus the proof of consistency is true also for the removal of datatype properties and semantic mappings. In turn, the functionality “Removing Datatype Properties, Bridging Connectors and Semantic Mappings” is validated for all the three property types.

Table 62. SPARQL query to prove consistency after deleting properties

```
SELECT DISTINCT ?predicate ?object
WHERE {
<http://fhnw.ch/modellingEnvironment/DSML4PTM#ICFStandardIsPartOfKoGuDo> ?predicate ?object .
}
```

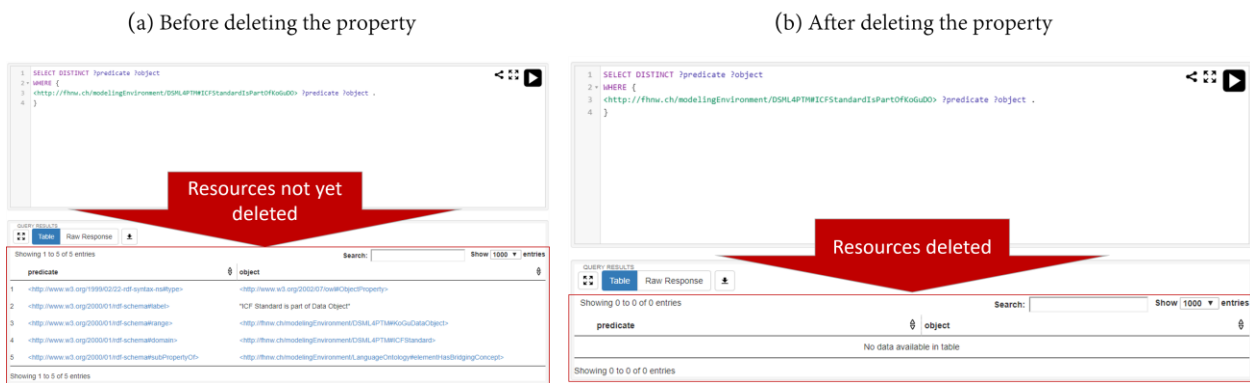


Figure 167. Query results (a) before and (b) after deleting the property

7.4.5 Validation of Functionalities for Editing Modelling Constructs and Properties

AOAME's functionalities "Editing Modelling Construct", "Editing Datatype Properties, Bridging Connectors and Semantic Mappings" are validated by implementing Use Case 5: Editing ICF Standard document and properties. In this case also, the use case is a follow-up of Use Case 3 (Sub-section 7.4.3). The same set of ontologies are, therefore, uploaded to the triplestore. The considered concepts and relations were already shown in Figure 162 of Sub-section 7.4.4.

7.4.5.1 Description of Use Case 5: Editing Modelling Element ICF Standard and Properties

In this use case the transferal manager is not satisfied with the properties of the ICF Standard document and suggests some changes. Specifically, the following properties should be edited:

1. the label, comment, graphical notation for the palette (i.e. annotation properties);
2. the bridging connector (i.e. object property);
3. a datatype property;

The annotation properties of the modelling element are edited by the functionality "Edit Modelling Construct", which updates resources in both the Modelling Language Ontology and the Palette Ontology. The remaining two properties are edited by the functionality "Editing Datatype Properties, Bridging Connectors and Semantic Mappings", which updates resources in the Modelling Language Ontology only.

The three following sub-sections contain the description of the user actions applied to the use case to change (1) annotation properties (Sub-section 7.4.5.1.1), (2) object properties (Sub-section 7.4.5.1.2) and (3) datatype property (Sub-section 7.4.5.1.3) of the modelling element "ICF Standard".

7.4.5.1.1 Editing Annotation Properties for ICF Standard

The following steps describe the user actions performed to change the annotation properties of the modelling element ICF Standard. The last two steps (6 and 7) are added to show that changes were applied. The steps are also depicted in Figure 169.

- Step 1: right-click on the "ICF International Standard" and click on "Edit ICF International Standard" modelling element. This action opens the Edit window, which shows all the annotation properties of the selected modelling element. The annotation properties are retrieved by the SPARQL SELECT, which is dynamically created. The upper part of Figure 168 shows the fragment of the query that retrieves the annotation properties. The bottom part of Figure 168 shows the query result with the annotation properties.


```

1 PREFIX po: <http://fhnw.ch/modelingEnvironment/PaletteOntology#>
2 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
3 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
4 PREFIX lo: <http://fhnw.ch/modelingEnvironment/LanguageOntology#>
5 PREFIX dsml4ptm: <http://fhnw.ch/modelingEnvironment/DSML4PTM#>
6 SELECT DISTINCT ?label ?comment ?graphicalNotation4Palette ?graphicalNotation4ModelEditor
7 WHERE {
8   ?modelingElement rdfs:label ?label .
9   ?modelingElement rdfs:comment ?comment .
10  FILTER (?modelingElement = dsml4ptm:ICFStandard) .
11  ?paletteElement po:paletteConstructHasPaletteThumbnail ?graphicalNotation4Palette .
12  ?paletteElement po:paletteConstructHasModelImage?graphicalNotation4ModelEditor .
13  FILTER (?paletteElement = po:ICFStandard) .
14 }

```

Retrieved annotation properties

QUERY RESULTS

Table Raw Response

Showing 1 to 1 of 1 entries Search: Show 1000 entries

label	comment	graphicalNotation4Palette	graphicalNotation4ModelEditor
1 "ICF International Standard"	"International Classification of Functioning, Disability and Health https://bioportal.bioontology.org/ontologies/ICF?p=classes"	"ICFstandard-dkm.png"	"ICFstandard-dkm.png"

Showing 1 to 1 of 1 entries

Figure 168. SPARQL query to show current annotation properties of ICF Standard

- Step 2: change the label and the comment. Additionally, the graphical notation for the palette is changed whereas the one for the model editor remains unchanged.
- Step 3: changes are completed.
- Step 4: click on the “Save” button to apply the new changes. This action instantiates SPARQL rule 8 (see Sub-section 5.3.4.4.4), which foresees a DELETE statement followed by an INSERT statement. Hence, the new changes is stored in the ontology.
- Step 5: (a) both the new graphical notation and the new label for the modelling element *dsml4ptm:ICFStandard* are displayed in the palette. Same as before the changes, the graphical notation appears only after the cursor is moved on the parent class “Data Document”. (b) By clicking on the new graphical notation, a new model element is instantiated in the Model Editor. Note that since the graphical notation for the Model Editor was not changed, the model element shows a different graphical notation than the palette element.
- Step 6: In order to demonstrate that the changes were applied, right-click on the modelling element “ICF International Standard” and select “Edit ICF Standard Document”.
- Step 7: All the changed properties are shown with the new values.

Differently from the previous validation approaches, in this use case the query result can be omitted as it would return the same annotation properties shown in step 7 of Figure 169.

In conclusion, changing annotation properties of a modelling construct leads to consistency between the graphical representation and the knowledge in the triplestore. Thus, the functionality “Edit Modelling Construct” is validated.

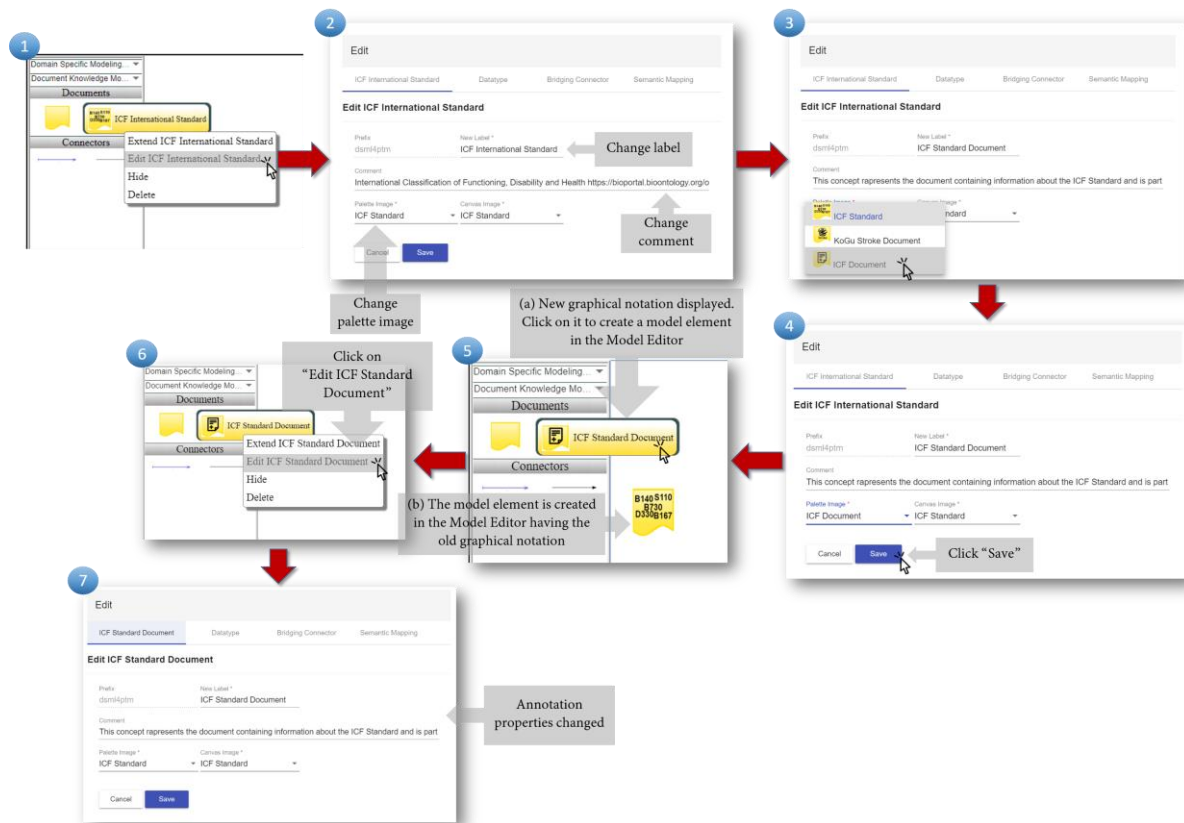


Figure 169. Steps to change annotation properties of a modelling construct on the GUI

7.4.5.1.2 Edit Object Properties for ICF Standard

Editing object properties of a modelling construct through the functionality “Editing Datatype Properties, Bridging Connectors and Semantic Mappings” consists of changing the label or the range of both bridging connectors and semantic mappings. The functionality is supported by dynamically instantiating SPARQL Rule 9 (see Sub-section 5.3.4.4.5). The SPARQL rule foresees a DELETE statement followed by an INSERT statement.

In this activity, the language engineer changes the label of the bridging connector “*dsml4ptm:ICFStandardIsPartOfKoGuDataObject*” and its range. The bridging connector is between the classes *dsml4ptm:ICFStandard* and *dsml4ptm:KoGuDataObject* (see also Figure 162 in Sub-section 7.4.4).

The required user actions to make such changes are described below. The steps are also depicted in Figure 171.

- Step 1: right-click on the modelling element “ICF International Standard” and click on “Edit ICF International Standard”, which opens the Edit window. The window shows all the properties of the selected modelling element.
- Step 2: select the “Bridging Connector” tab, which opens the window to edit the bridging connector.

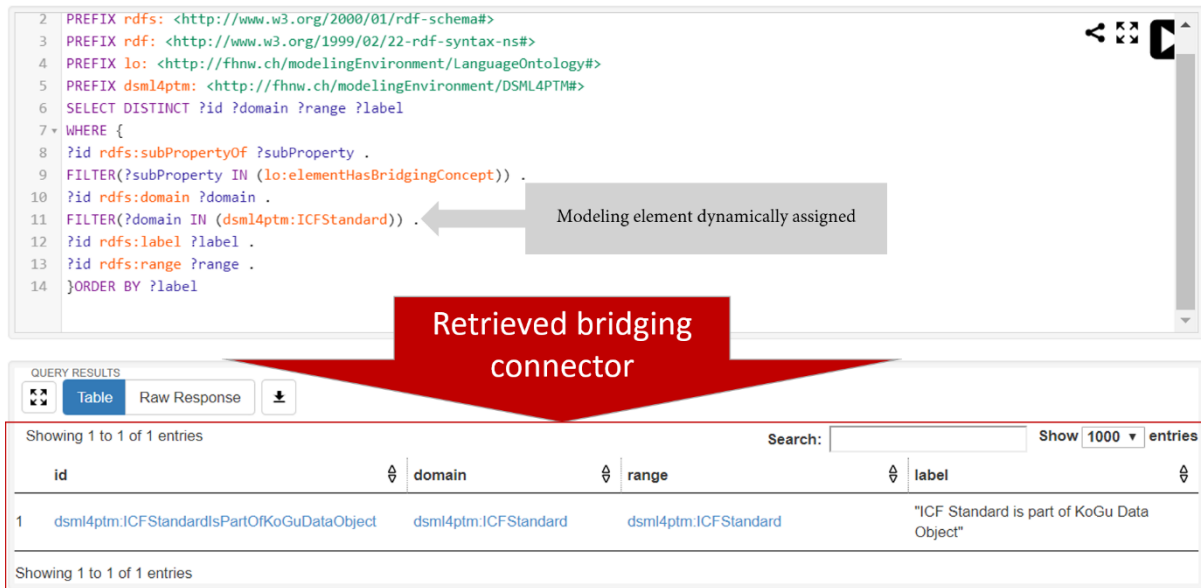


Figure 170. Result of the dynamically generated SPARQL SELECT that retrieves bridging connectors

- Step 3: the bridging connector *isPartOf* is retrieved by the SPARQL SELECT, which is dynamically generated from the endpoint *getBridgeConnectors* (see Sub-section 6.3.2.1). The upper part of Figure 170 shows the generated query. Note that since we are editing the modelling element *dsml4ptm:ICFStandard*, it is dynamically assigned to the query instance. Bottom Figure 168 shows the result of the query, with the bridging connector *dsml4ptm:ICFStandardIsPartOfKoGuDataObject*. Therefore, the bridging connector is displayed in the view. Click on the drop-down list of the bridging connector, select the connector *isPartOf*, and click on “Modify”. The action opens a new window “Edit Relation”.
- Step 4: from the “Edit Relation” window, change the label into “ICF Standard is part of Data Object” and the range to “bpmn:DataObject”. The new range can be retrieved through the search box, which (in this case) queries all the subclasses of *lo:ModellingElement*.
- Step 5: this view shows the new label and range for the bridging connector.
- Step 6: the new label and range for the bridging connector are saved and stored into the triplestore. Same as in step 2, the bridging connector is retrieved by querying the ontology.

Step 6 validates the functionality for the update of bridging connectors. Hence, the query result from the triplestore is omitted as it would return the same bridging connector shown in Step 6 of Figure 160. The graphical- and machine-interpretable representation is, therefore proved.

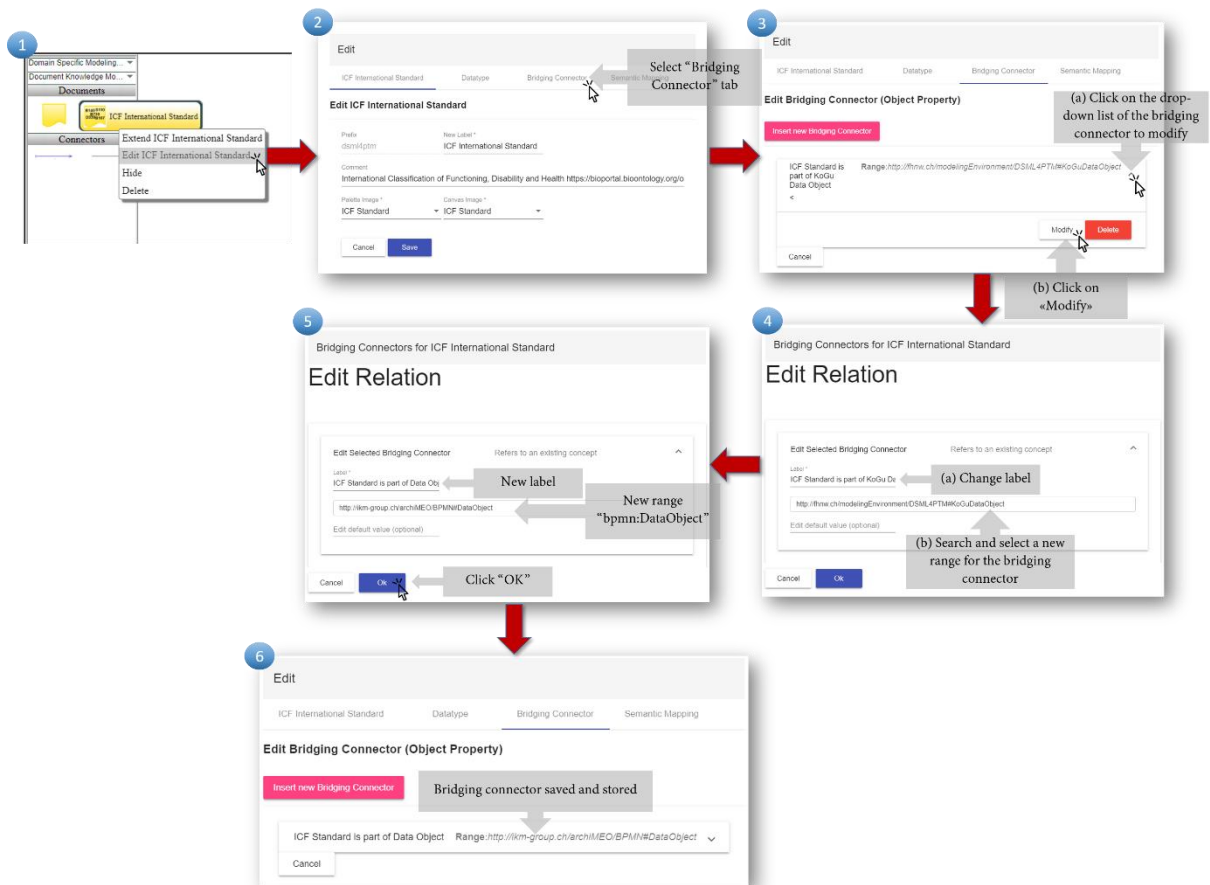


Figure 171. Steps to change bridging connectors of a modelling construct on the GUI

Changes performed on the semantic mappings require the same steps as the described changes of bridging connectors. The only difference concerns the SPARQL query for the retrieval of the object properties. Specifically, the query from the Semantic Mapping tab, retrieves the sub-properties of *lo:elementIsMappedWithDOconcept*, instead of *lo:elementHasBridgingConcept* (see row 9 on top of Figure 170). Figure 172 shows the steps required to change the semantic mapping *ICFStandardIsMappedWithBodyFunction*. As when editing a bridging connector, step 6 of Figure 172 shows the changed semantic mapping (see range *icf:ICFQualifier* in step 6 of Figure 172), which is retrieved from the triplestore. Therefore, Step 6 validates the changes for the semantic mappings. Likewise, the consistency between the graphical and machine-interpretable representation of the modelling language is preserved.

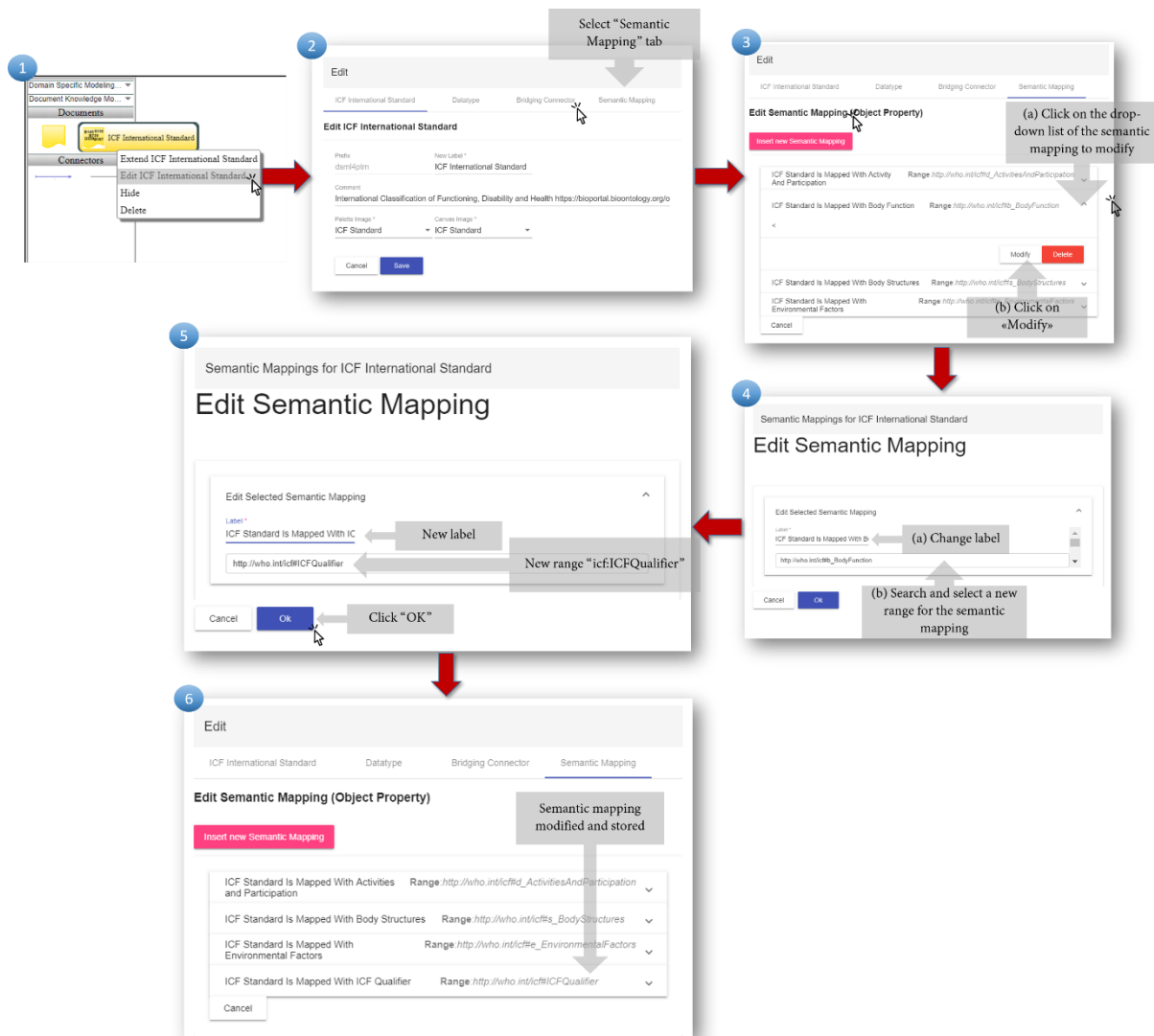


Figure 172. Steps to change semantic mappings of a modelling construct on the GUI

7.4.5.1.3 Validation of Functionality for Editing Datatype Properties

Editing datatype properties of a modelling construct consists of changing the label and value type, e.g. *xsd:integer*, *xsd:string*, *xsd:dateTime* etc. The functionality the functionality “Editing Datatype Properties, Bridging Connectors and Semantic Mappings” is also supported by SPARQL rule 10 (see Sub-section 5.3.4.4.6). The SPARQL rule foresees a DELETE statement followed by an INSERT statement.

In the considered edit activity, the language engineer changes the datatype property *dsml4ptm:ICFStandardhasTimeStamp*, which connects the modelling elements *dsml4ptm:ICFStandard* with the value type *xsd:dateTime*. In particular, the label of the datatype property and the range are to be changed into “ICF Standard Document has Day of Creation” and “*xsd:date*”, respectively.

The user actions to make such changes are described below. The steps are also depicted in Figure 174.

- Step 1: right-click on the modelling element “ICF International Standard” and click on “Edit ICF International Standard”. This action opens the Edit window, which shows all the properties of the selected modelling element.
- Step 2: Select the “Datatype” tab, which opens the “Edit Datatype Property” window.

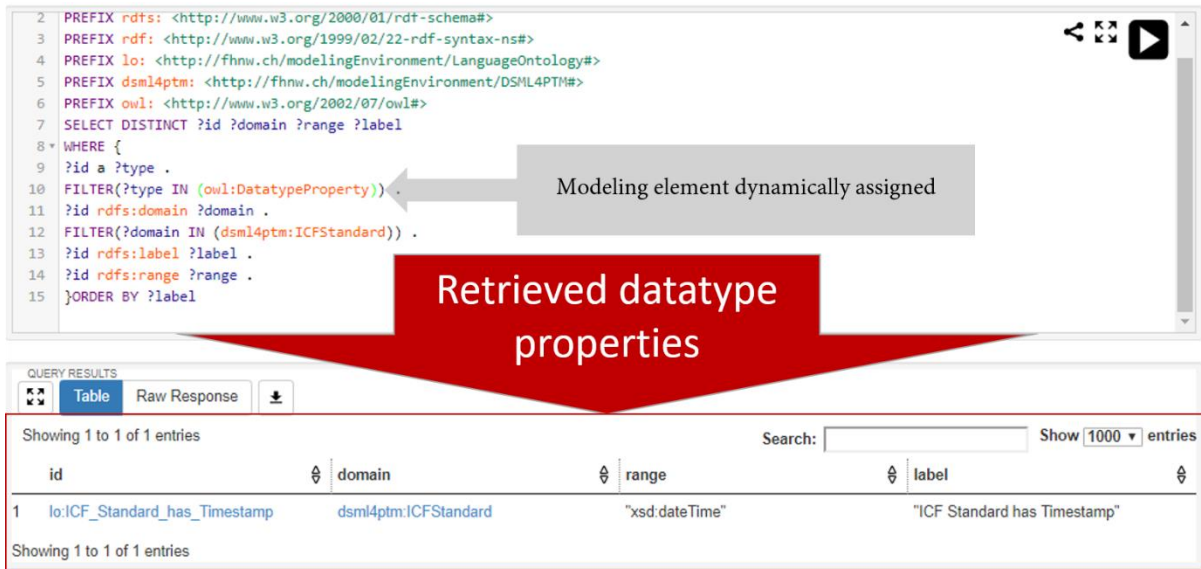


Figure 173. Result of the dynamically generated SPARQL SELECT that retrieves datatype properties

- Step 3: as soon as the “Edit Datatype Property” window opens, the datatype property *hasTimeStamp* is retrieved by the SPARQL SELECT, which is dynamically generated from the endpoint *getDatatypeProperties* (see Sub-section 6.3.2.1). The top of Figure 173 shows the generated query. Note that since we are editing the modelling element *dsml4ptm:ICFStandard*, this is dynamically assigned to the query. Bottom of Figure 173 shows the result of the query, which contains the datatype property labelled as “ICF Standard has Timestamp”. This datatype property is shown in the view of Step 3 (see Figure 174). Click on the drop-down list of the datatype property, select the property “ICF Standard has Timestamp” and click on the “Modify” button. Subsequently, the “Edit Attribute” window opens.
- Step 4: from the “Edit Attribute” window, change the label and the value type of the datatype property, i.e. “ICF Standard Document has Day of Creation” and *xsd:date*, respectively.
- Step 5: click on the “OK” button to store the new values in the ontology.
- Step 6: the new label and datatype are displayed in the “Edit Datatype Property” window”. Same as in step 2, the datatype property is displayed as it is retrieved from the triplestore.

Step 6 validates the changes for the datatype properties. Hence, the query result from the triplestore is omitted as it would return the same datatype property shown in step 6 of Figure 174. Consistency between the graphical and machine-interpretable knowledge is proved. Therefore, the functionality “Editing Datatype Properties, Bridging Connectors and Semantic Mappings” is validated also for the change of datatype properties.

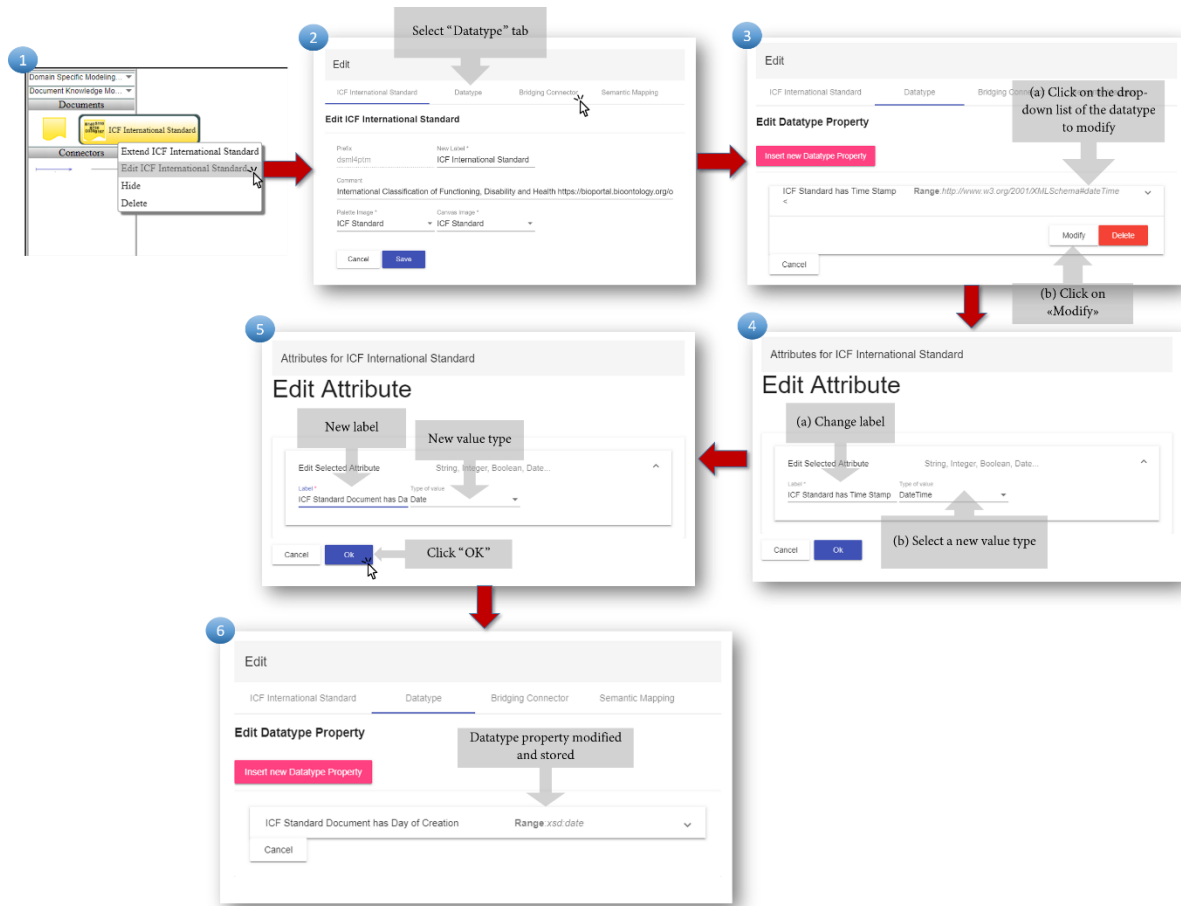


Figure 174. Steps to change datatype properties of a modelling construct on the GUI

7.4.6 Validation of Functionality for Hiding Modelling Constructs

This sub-section describes the validation of the functionality for “Hiding Modelling Constructs” by implementing Use Case 6. Like in the previous sub-section, the use case refers to the Patient Transferal Management application domain and presents the same concepts and relations depicted in Figure 162 of Sub-section 7.4.4.

In this use case the language engineer hides the modelling element ICF Standard to simplify the look of the palette. This functionality “Hide Modelling Construct” is supported by the dynamic instantiation of SPARQL Rule 11 (see Sub-section 5.3.4.4.7). This SPARQL rule foresees a DELETE statement followed by an INSERT statement.

The user actions are depicted in Figure 174 and described in the following two steps:

- Step 1: right-click on the modelling element to hide “ICF International Standard”. Next, click on “Hide”.
- Step 2: the hidden modelling element no longer appears in the palette.

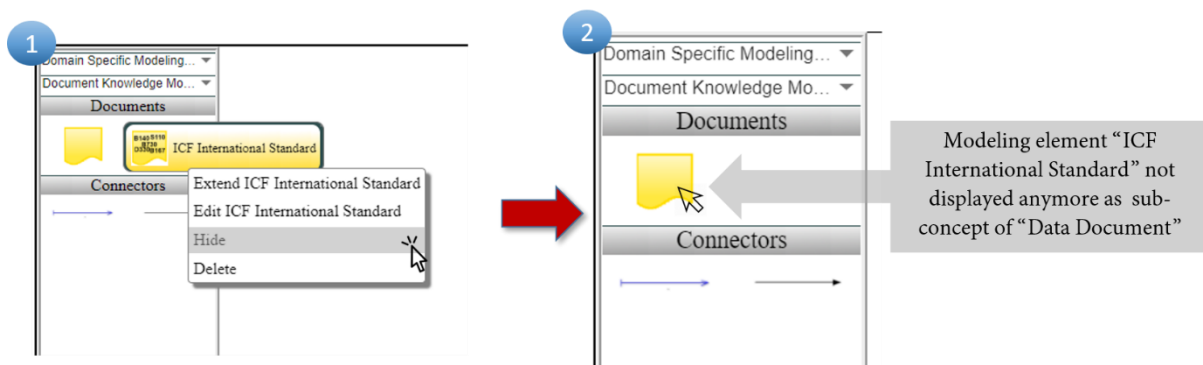


Figure 175. Steps to hide a modelling construct from the palette

7.4.6.1 Query Result After Implementing Use Case 6 - Hiding ICF Standard

The generated instance of SPARQL Rule 11 ensures that the change is consistently propagated to the ontology. In order to prove that, the query in Table 63 was fired against the ontology in the triplestore both before and after hiding the modelling element. Specifically, the SPARQL query retrieves the Boolean value, which is associated to the datatype property *po:paletteConstructIsHiddenFromPalette* of the hidden modelling element ICF Standard.

Both query results are depicted on bottom of Figure 176. Results on the left-hand side of the figure show the “*false*” Boolean value, whereas on the right-hand side the “*true*” Boolean value is shown.

In conclusion, the functionality proves to preserve consistency between the graphical and the machine-interpretable knowledge. Therefore, the functionality “Hide modelling construct” is validated.

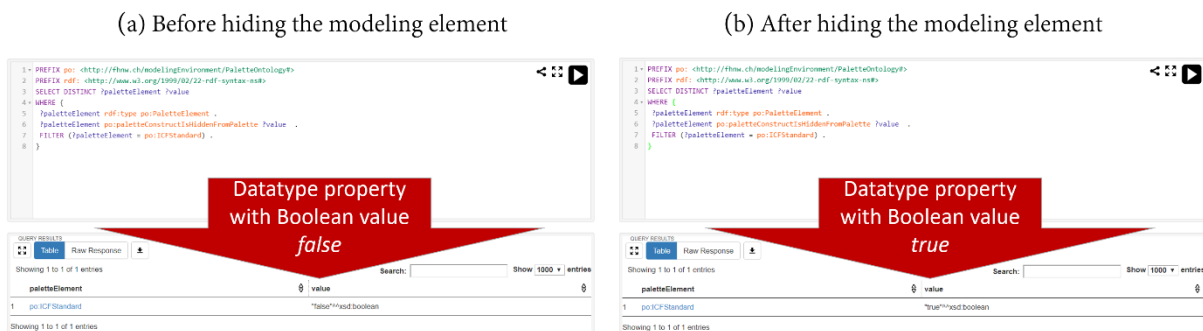


Figure 176. Query results (a) before and (b) after hiding a modelling element from the Palette

Table 63. SPARQL query to retrieve the value of a datatype property

```

PREFIX po: <http://fhnw.ch/modellingEnvironment/PaletteOntology#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT DISTINCT ?paletteElement ?value
WHERE {
  ?paletteElement rdf:type po:PaletteElement .
  ?paletteElement po:paletteConstructIsHiddenFromPalette ?value .
  FILTER (?paletteElement = po:ICFStandard) .
}

```

7.4.7 AOAME for Innovation Processes

In order to further prove the generality of the approach, the agile and ontology-aided meta-modelling approach was implemented to support one of the most widespread innovation processes: Design Thinking. Design Thinking is a five-phase innovation process (empathies, define, ideate, prototype and test), where sketches are used from an early phase to increase problem understanding, as well as later on to ideate the new solution designs (Institute of Design Stanford, 2010). The sketching activity is being recently supported with pre-defined digital modelling elements called SAP Scenes (Miron, Muck, & Karagiannis, 2019). However, the set of modelling elements is limited, and the elements are not sufficiently expressive to address all possible application domains.

To overcome this problem, the set of modelling elements SAP Scenes were integrated in the agile and ontology-aided meta-modelling approach (Laurenzi et al., 2020). Thus, new graphical elements and related domain-specific aspects are accommodated on-the-fly as they are needed. The approach allows modelling both unforeseen scenarios and domain-specific scenes for which the available elements are not expressive enough.

The integration of the SAP Scenes required their transformation into an ontology format, i.e. SAP Scenes Ontology, which is then implemented in the ontology architecture of the AOAME prototype.

To demonstrate the applicability of the approach in Design Thinking, Figure 177 shows a sequence applied in a kitchen scenario. Step 3 shows the sketch of a desired kitchen. The dashed red arrow points to the modelling element “double door fridge”, which was missing at the time of modelling. Thus, the new fridge type was added on-the-fly through the AOAME’s functionality “Extending Modelling Construct” (see Step 1 and Step 2 of Figure 177).

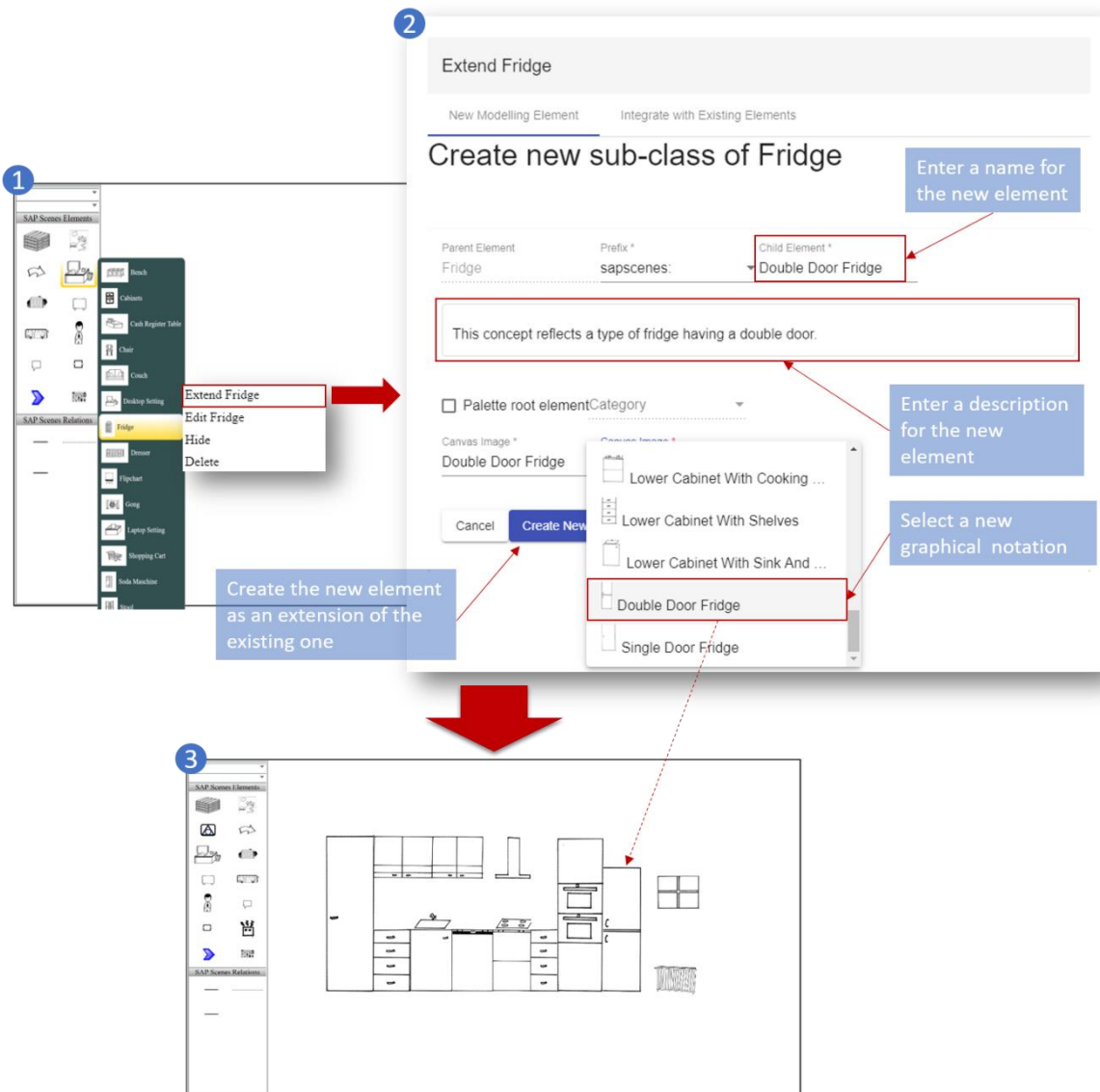


Figure 177. The agile and ontology-aided met-modelling approach to support Design Thinking

7.5 Evaluation Conclusion and Considerations

The evaluation followed the aspects determined in the evaluation strategy, which are summarised in Table 56 (Section 7.1). The building process of the evaluation strategy was supported by the framework of Pries-Heje et al., (2008). Here, I have discussed the three main dimensions, which refer to the questions: *what to evaluate* (i.e. the agile and ontology-aided meta-modelling approach), *when to evaluate* and *how to evaluate*. Additional literature is considered in the “when” and “how” dimensions. In the former, it contributed to suggest the adoption of the *prototyping* pattern as an *evaluation method*, which was based on the identified evaluation activity *Eval3*. The latter is performed after the implementation of the artefact and aims to identify *how well* the artefact performs in providing a solution to the addressed problem. The prototype evaluation method is appropriate for both *artificial* and *naturalistic* evaluation forms. The evaluation forms are discussed in the “how to evaluate” dimension. The forms narrowed the evaluation purpose to the *utility* and *scope* (naturalistic), and *correct design of the artefact* (artificial). The illustrative scenario method was then added as an additional evaluation method to prototype for untidily and scope of the artefact. The evaluation purposes were considered together with the artefact type (i.e. method) to determine the evaluation criteria, namely: (1) *the extent to which requirements are satisfied by the implemented approach*, its (2) *operationability* and (3) *generality* of the approach. Thus, a qualitative and positivistic evaluation of the approach was performed with respect to these criteria.

Regarding the first criteria (*the extent to which requirements are satisfied by the implemented approach*), the elaboration on Section 7.3, demonstrates how all eight requirements are fulfilled by AOAME’s functionalities. For this, each of AOAME’s functionalities reports which of the 11 SPARQL rules are incorporated as well as which of the 10 operators are addressed. Since all requirements are completely fulfilled by AOAME, it can be asserted that the evaluation on the correct design of the approach is successfully carried out.

For evaluating the utility of the approach, the following criteria were considered:

- Operationability of the approach: The ability of the approach to preserve consistency between the graphical and the machine interpretable representation while performing on-the-fly domain-specific adaptations of modelling languages.
- Generality of the approach: The ability of the approach to be applied in different application domains.

The operationability of the approach was evaluated by implementing real-world use cases in AOAME, through its numerous functionalities. All the functionalities demonstrated a preserved consistency between the graphical and the machine-interpretable representation of modelling languages while performing domain-specific adaptations. Hence, the operationality of the approach is proved.

The generality of the approach was also proved as AOAME was successfully applied in different application domains, namely: the patient transferal management, business process as a service and in the innovation process Design Thinking.

The evaluation activities for the correct design and operationability provided feedback to refine the agile and ontology-aided meta-modelling approach. Specifically, the evaluation activities supported the refinement of the operators, semantic rules and their relationship. Additionally, issues concerning technical aspects were improved as well. These were (a) the correct generation of SPARQL rules from the implemented methods (b) the correct query results of the search functionalities and (c) the look and feel of the GUI. Each point is detailed below.

- a. There have been cases where methods were not producing the wanted SPARQL rule and, thus, amendments were applied.

- b. A similar issue was reported for the queries of search functionalities, in which expected results might not have been shown. For instance, the query that implements the creation of the semantic mappings required changes.
- c. The look and feel of the GUI, on the other hand, was improved for a more intuitive distribution of meta-modelling activities (e.g. changing or adding elements, graphical notations, labels, attributes or relations). In particular, the sequential windows for the meta-modelling were created to address presented information in a step-wise approach. For instance, in the extension functionality the first window has annotation properties and graphical notations whereas properties only appear in the subsequent window. Input for the style improvements also came when exposing AOAME in conferences and, research visits, as well as to colleagues and masters students. Further improvements can be identified by performing usability evaluation of AOAME, which is out of scope of this research, thus is left to future work.

These improvements demonstrate the effectiveness of the feedback loop of the Design Science Research from the evaluation to the design cycle of the artefact.

Finally, the successful evaluation has demonstrated that the proposed agile and ontology-aided meta-modelling approach is capable of tackling the two main challenges presented in Section 4.3, i.e.:

Challenge #1: An agile meta-modelling approach should promote the tight cooperation between domain experts and language engineers by avoiding the separation between the language engineering and modelling components.

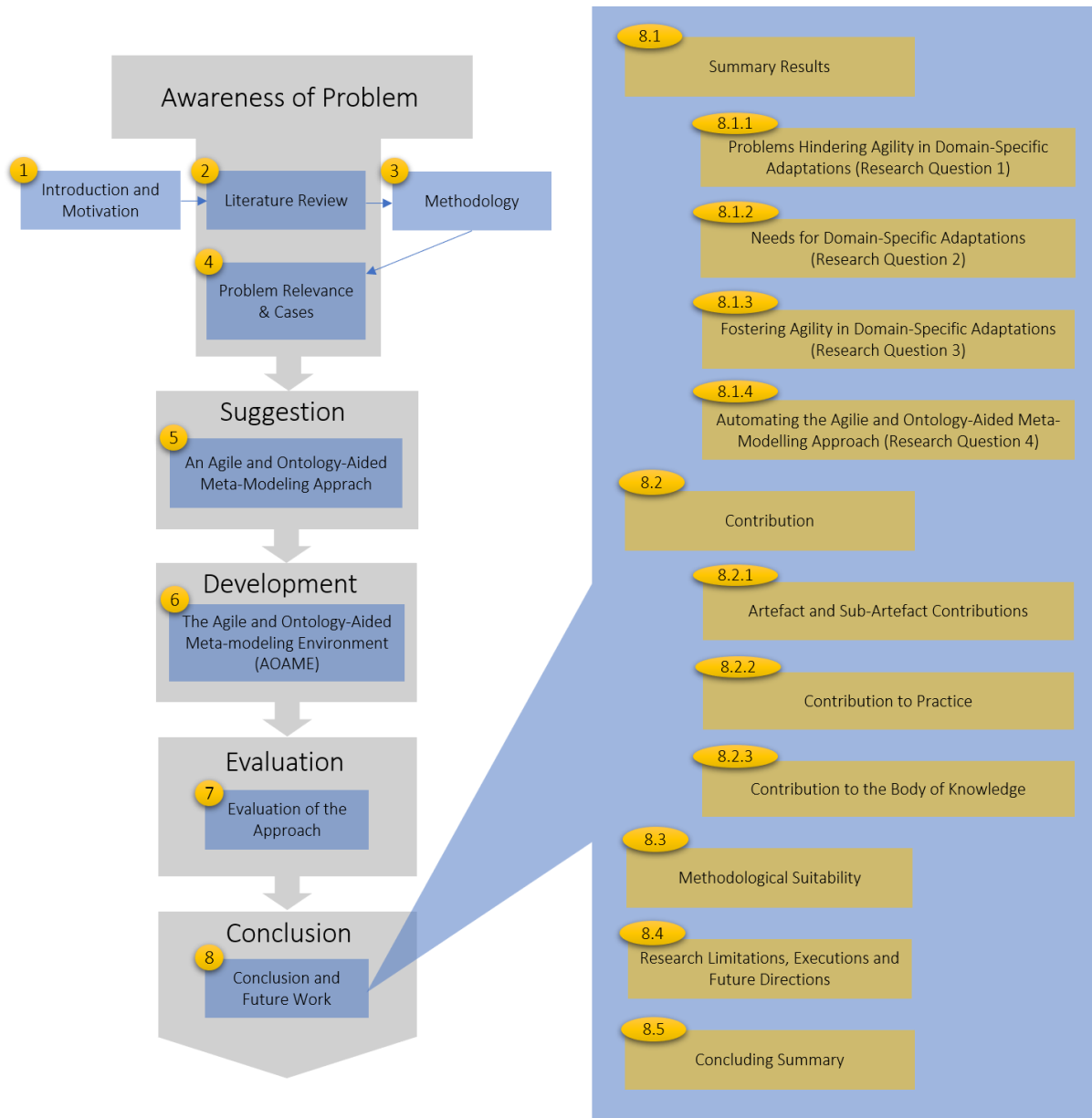
The first challenge is tackled by integrating language engineering and modelling into one single component. The integrated component allows the interleave between language engineering, modelling and evaluation activities. Therefore, the tight cooperation between language engineers and domain experts is promoted from an early stage of DSML engineering.

Challenge #2: An agile meta-modelling approach should avoid sequential DSML engineering phases while adapting modelling languages.

The second challenge is tackled by the possibility to perform domain-specific adaptations of modelling languages on-the-fly. As changes are suggested, they can be accommodated immediately, through the AOAME's functionalities. Therefore, the new requirements do not have to go through the sequential language engineering phases before a new version of the DSML is released for usage.

A getting started guide to use AOAME can be found in Appendix E: Prototype , folder E6.

8. CONCLUSION AND FUTURE WORK



The research conducted in this work is within the Information Systems (IS) discipline and conceives a new scientific agile and ontology-aided meta-modelling approach. It combines meta-modelling of Domain-Specific Modelling Language (DSML) for Enterprise Modelling (EM) with the use of Knowledge Representation and Reasoning (KRR), which is a subset of Artificial Intelligence. For Knowledge Representation and Reasoning (KRR), ontologies and semantic rules are used.

Meta-Modelling is an approach to develop Domain-Specific Modelling Languages (DSML) in Enterprise Modelling (EM). This research focused on the meta-modelling practice of re-using existing modelling languages to develop DSMLs, i.e. *domain-specific adaptations of modelling languages*. The literature review focused on raising awareness of approaches for domain-specific adaptations of modelling languages. Moreover, two cases from the practice of domain-specific adaptations were analysed. Expert interviews were conducted as an additional input to ensure relevance to the practice of domain-specific adaptation in industry. As a result,

a list of requirements was elicited to create an agile meta-modelling approach, which is theoretically grounded as well as relevant in application domains.

It was shown, however, that existing approaches would lead to inconsistency issues when performing domain-specific adaptations of modelling languages. Inconsistency would arise between the graphical and the machine-interpretable representations of modelling languages. To tackle this problem, the agile approach was incorporated into an ontology-aided approach. The latter preserves machine-interpretable representation of modelling languages also when changes occur. As a result, an agile and ontology-aided meta-modelling approach was proposed. Next, the approach was implemented in a prototypical modelling environment called AOAME.

The main contributions of this research work are summarized in the following three points:

1. An agile meta-modelling approach allows intertwining language engineering, modelling and evaluation activities quickly. For this, language engineering and modelling were integrated in a single component. The latter enables the on-the-fly domain-specific adaptations of modelling languages. Ten operators were derived to support such adaptations. The approach has two prominent benefits:
 - a. it promotes the tight cooperation between the language engineer and the domain expert from an early stage of the DSML engineering;
 - b. it avoids the sequential engineering phases while engineering a DSML.

The two benefits foster the quick and high-quality development of DSMLs. This contribution tackles the first research problem about the lack of agility in domain-specific adaptations of modelling languages.

2. An ontology-aided approach preserves consistency between the graphical and the machine-interpretable representation of modelling languages when performing the domain-specific adaptations on-the-fly. Therefore, it tackles the second research problem. In particular, the specification of a modelling language is defined in an ontology, which allowed creating a machine-interpretable representation for the corresponding graphical modelling languages. In order to preserve consistency while domain-specific adaptations occur, eleven semantic rules were proposed in support of the ten operators.
3. The agile and ontology-aided meta-modelling approach was implemented in the modelling environment AOAME. Functionalities are implemented allowing the user to perform the on-the-fly domain specific adaptations of modelling languages. SPARQL rules are dynamically generated to propagate changes to the ontology. Therefore, the preservation of consistency between the graphical and machine-interpretable representations of modelling languages was automated.

The remainder of this chapter is structured as follows: in the next section, results are summarized with respect to the research questions and references to the relevant chapters and sections in the body of the thesis will be provided. Next, the contributions of this research work are discussed in terms of (1) artefacts (2) practice, and (3) body of knowledge (Section 8.2). The methodological suitability of the artefact is reported in Section 8.3. Then research limitations and future work are listed in Section 8.4. Finally, Section 8.5 contains the concluding summary.

8.1 Summary Results

This section presents a summary of the thesis results by providing the answers to the research questions.

As already mentioned in Section 1.2, this research work followed the constructive approach suggested by Cronje (2011) so that the answers to research questions contribute to answer the main research question. Hence, the answers described in the following sub-sections contribute to answering the main research question:

How can agility for domain-specific adaptations of modelling languages be fostered while preserving seamless consistency between their graphical and machine-interpretable representations?

8.1.1 Problems Hindering Agility in Domain-Specific Adaptations (Research Question 1)

Research question 1 (*RQ1*) guided the research towards identifying problems that hinder agility of domain-specific adaptations.

RQ1: What are the problems that hinder agility in domain-specific adaptations of modelling languages?

To answer this research question, understanding was increased on how to perform domain-specific adaptations of modelling languages. The case study research strategy was employed, from which two cases were created. In each case one domain-specific modelling language (DSML) was developed through domain-specific adaptations (Sections 4 and 4.2).

From existing work and literature, it was identified that DSMLs are created either for software development (see Sub-section 2.6.1) or for knowledge retention (see Sub-section 2.6.2). This research work focused on the latter. DSMLs for knowledge retention has the ultimate goal to support decision making. To achieve this goal, one way is to create a DSML with the solely declarative purpose, i.e. domain-specific aspects of an application domain are modelled for human interpretability. An alternative is to extend the declarative purpose to an automation purpose, where model aspects are machine-interpretable. The first modelling purpose falls within the field of Knowledge Management while the second one extends to the field of Knowledge Engineering.

The first case on *Patient Transferal Management* (Section 4) relates to Knowledge Management, which foresees only the declarative representation of relevant domain-specific aspects. The other case *Business Process as a Service* (Section 4.2) deals with Knowledge Engineering, which foresees not only a declarative representation of domain-specific aspects, but also automation of knowledge using ontologies.

In the case studies two domain-specific modelling languages (i.e. DSML4PTM and BPaaS DSML) were created following the AMME Lifecycle methodology (Karagiannis, 2018). The methodology instantiates the Agile Modelling Method Engineering framework (Karagiannis, 2015) and provides a rigorous methodology for the development of DSMLs. From existing work and literature (see Sub-section 2.10.4) it was identified that this methodology is the most suitable for domain-specific adaptations of existing enterprise modelling languages.

Both approaches were implemented in the ADOxx toolkit. For BPaaS, the approach included the semantic lifting of models, which was implemented through an ADOxx functionality. The functionality allowed to annotate models with ontology concepts. Subsequently, the BPaaS DSML was implemented using an ontology-based meta-modelling and the results were shown in and validated through a web-based interface.

The two cases were analysed (see Section 4.3) to answer RQ1. From this, a list of problems hindering agility of DSML engineering was elaborated together with the two main challenges to tackle. The two main challenges revert to the following two agile principles:

- the tight cooperation between the language engineers and the domain experts while engineering DSMLs;
- the avoidance of sequential engineering phases while engineering DSMLs.

8.1.2 Needs for Domain-Specific Adaptations (Research Question 2)

Research question 2 (RQ2) guided the research to increase understanding of the needs for domain-specific adaptations of modelling languages.

RQ2: What are the needs for domain-specific adaptations of modelling languages?

In order to provide an answer, interviews were performed with modelling experts from various companies (see Section 4.4). The analysis of the interview results led to the derivation of three main adaptation complexity levels: (1) modelling language simplification, (2) change of abstract syntax and notation and (3) extend abstract syntax and add notation. From level 1 to level 3 the complexity for domain-specific adaptations increases. Finally, interview findings were combined with the literature and lessons learned from the two DSMLs development to derive a list of requirements for the domain-specific adaptations. The requirements for domain-specific adaptations allow the following:

- The simplification of modelling languages, which includes the removal of modelling constructs and properties as well as hiding modelling constructs.
- The change of modelling constructs, which includes the update of properties of the modelling constructs.
- The extension of modelling constructs, which includes the creation of new modelling constructs and properties.
- The integration of existing modelling constructs, which is done by creating the sub-class property among modelling constructs from different modelling languages.

8.1.3 Fostering Agility in Domain-Specific Adaptations (Research Question 3)

Research question 3 (RQ3) guided the research in the suggestion of an approach that fosters agility in performing the domain-specific adaptations of enterprise modelling languages.

RQ3: How can agility be fostered when performing domain-specific adaptations of modelling languages?

This research question is answered by suggesting an approach for the agile and ontology-aided meta-modelling, which is described in Chapter 5. The approach addresses the two main challenges derived in the answer of RQ1 as well as fulfils the requirements elicited in the answer of RQ2. Namely, the new agile meta-modelling approach

- promotes the tight cooperation of the language engineering and the domain experts from an early stage of DSML engineering by allowing the intertwine of language

engineering, modelling and evaluation activities. For this purpose, language engineering and modelling were integrated into one component (Section 5.1);

- avoids the sequential DSML engineering phases by allowing the on-the-fly domain-specific adaptations in the integrated component.

The on-the-fly domain-specific adaptations are applied to affect the three main specifications of a modelling language: graphical notation, abstract syntax and semantics. A further literature investigation (Sub-section 5.1.4) has shown that language semantics not only resides in the abstract syntax (i.e. meta-model and constraints) but also in domain concepts that are language-independent. Hence, the seminal works in (Harel & Rumpe, 2000; Karagiannis & Kühn 2002; Atkinson & Kuhne, 2003) were considered to further specify the semantics with both semantic domain and semantic mapping and to distinguish between the linguistic and domain views of a language specification. An additional set of four requirements was thereby derived to focus on the domain-specific adaptations of semantic domain and semantic mappings.

All the requirements (eight in total) were used to derive a set of ten operators, which enable the domain-specific adaptations on-the-fly (Sub-section 5.1.6). The operators were conceived through several steps until a refined version was provided. A critical reflection on the side effects produced using the operators is also provided.

Since the side effects can be addressed by grounding a modelling language with machine-interpretable semantics, the approach is then elaborated in Section 5.2. The elaboration is supported by additional literature as well as by knowledge gained from creation of the semantically-enhanced BPaaS DSML (Section 4.2). From this elaboration it was understood that current approaches for grounding modelling languages with machine-interpretable semantics (such as semantic lifting) are not suitable for the proposed agile meta-modelling approach. That is, on-the-fly domain-specific adaptations of modelling language (i.e. human-interpretable representation) create inconsistencies with the reflecting ontology (i.e. machine-interpretable representation). Inconsistency problems are described in Sub-section 5.2.2.

To overcome these problems, an ontology-aided approach for agile meta-modelling was proposed and is described in Section 5.3. The approach builds upon an ontology-based meta-modelling approach. Conversely to semantic lifting, the ontology meta-modelling approach foresees the definition of the afore-mentioned modelling language specifications directly in the ontology. Therefore, there is no longer the need of annotating meta-models with ontology concepts. A new ontology architecture was conceived to support the definition of modelling language specifications in an ontology-based meta-model. The chosen ontology language is also described and motivated (see Sub-section 5.3.2.2). A set of eleven semantic rules were conceived to aid the propagation of changes from the graphical to the machine-interpretable representation of a modelling language. The rigorous and sound creation of the rules is ensured by following the methodology of Grüninger and Fox (1995), which is slightly adapted to create and validate SPARQL rules, syntactically and semantically. Each rule employs at least one operator, which allows the preservation of consistency between the graphical and machine-interpretable representations of a DSML while performing domain specific adaptations on-the-fly. As a result, the agile and ontology-aided meta-modelling approach answers research question 3.

8.1.4 Automating the Agile and Ontology-Aided Meta-Modelling Approach (Research Question 4)

Research question 4 (RQ4) guided the research in automating the proposed agile and ontology-aided meta-modelling approach.

RQ4: How can the agile approach for domain-specific adaptations that preserves seamless consistency between the graphical and the machine interpretable representation be automated?

The agile and ontology-aided meta-modelling approach is implemented in the prototypical modelling environment AOAME, i.e. **A**gile and **O**ntology-**A**ided **M**eta-modelling **E**nvironment. In particular, the approach is implemented in a Model-View-Controller (MVC)-based architecture, which consists of three components: a triplestore, a graphical user interface (GUI) and a web service. The three components and their technical specifications are described in Chapter 6.

The GUI has two sub-components: the palette and the model editor. The elaboration of the model editor is left to future work. At present, the model editor displays the graphical notations that are instantiated from the palette. Thus, graphical models can be created in the model editor. The palette is further elaborated in Section 6.2. It allows the on-the-fly domain-specific adaptations of modelling languages. First, the three-step approach implemented to display the graphical notations of modelling languages is introduced. This includes the explanation of the set of ontologies to upload on the triplestore. The set of ontologies reflects the specification suggested in the agile and ontology-aided meta-modelling approach. To realize the logic for displaying graphical notations in the palette, methods for the dynamic generation of SPARQL queries were implemented in the web service.

Section 6.3 describes the main functionalities that are implemented in AOAME and enable the on-the-fly domain-specific adaptations. Each functionality incorporates calls for the methods that dynamically instantiate the previously-mentioned eleven SPARQL rules. In this sense, the functionalities automate the propagation of changes from the graphical to the machine-interpretable representation of a modelling language. Therefore, while performing on-the-fly domain-specific adaptations, consistency between the two knowledge representations is ensured seamlessly.

8.2 Contribution

Artefacts (i.e. constructs, models, methods, and instantiations) are purposeful and built to address unsolved problems (Hevner et al., 2004).

The *main artefact* produced in this research work is an approach that fosters agility in domain-specific adaptations of modelling languages (or more in general terms meta-modelling) while preserving consistency between the graphical and machine-interpretable representation of the modelling languages. The approach is implemented in a prototypical modelling environment (AOAME).

The creation of the artefact was guided by the research questions. Several sub-artefacts were created by answering each of the research questions. The contributions of both artefact and sub-artefacts as well as their relationship is further elaborated in Sub-section 8.2.1.

The research work was set within a real-world context from the beginning. Use cases from two different application domains were implemented: (1) Patient Transferal Management from the eHealth domain and (2) BPaaS from the Cloud domain. In turn, the results of the research work are relevant to business practice. Moreover, results were recently transferred into innovation processes (Laurenzi et al., 2020).

8.2.1 Artefact and Sub-Artefact Contributions

According to Hevner and Chatterjee (2010) “the fundamental principle of design science research is that knowledge and understanding of a design problem and its solution are acquired in the building and application of an artefact”. In this research work, knowledge and understanding in each sub-artefact were gained incrementally and contributed to build the main artefact.

As depicted in Figure 178, the main artefact is specified in three main contributions: (1) an agile meta-modelling approach, (2) an ontology-aided meta-modelling approach, and (3) the AOAME prototype. The lower part of Figure 178 shows the five sub-artefacts. The dotted arrows show the knowledge flow among the (sub)artefacts contributions. It starts from the (sub)artefact where knowledge is gained and ends in a (sub)artefact where the gained knowledge is used as input.

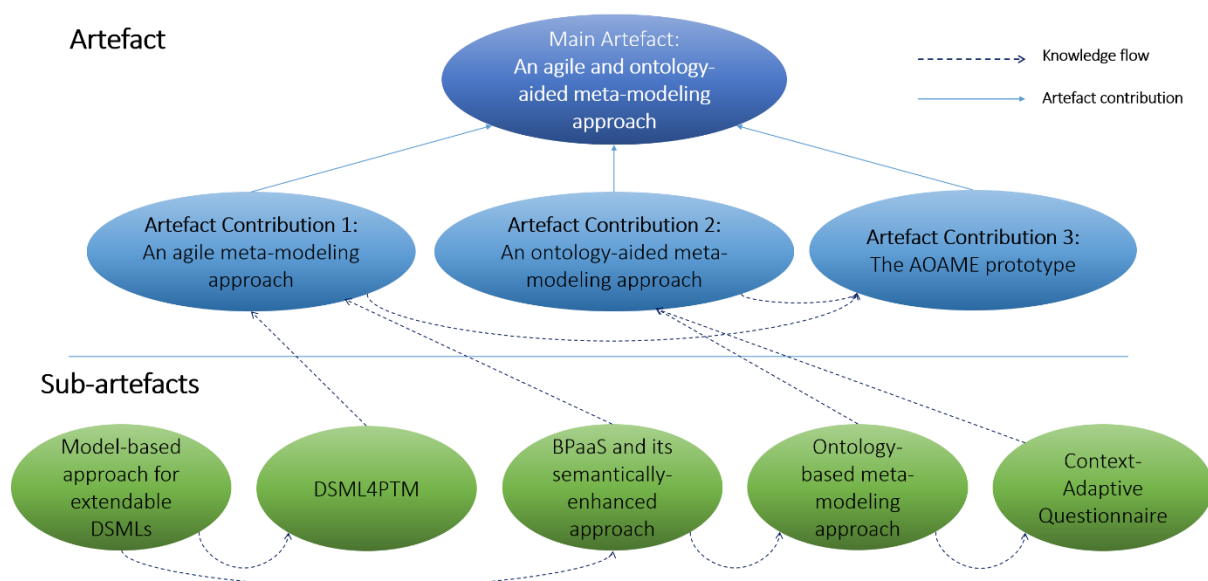


Figure 178. Main artefact, sub-artefacts and knowledge flow among them

Table 64 and Table 65 contain a comprehensible description of the three artefact contributions and the five sub-artefacts contributions, respectively. In both tables, each (sub)artefact contribution is specified with (a) a description, (b) the unsolved problem, (c) what knowledge was gained, whether it was gained from the conceptual and/or from the practical understanding, the knowledge flow, and (e) the reference(s) in the thesis.

Table 64. Main artefact contribution

<i>Artefact Contribution</i>	Description	Unsolved Problem	Gained Knowledge	Reference in the thesis
<i>Agile meta-modelling approach.</i>	An approach for the on-the-fly domain-specific adaptations of modelling languages. It includes (1) the integration between the language engineering and modelling components in a single modelling environment; (2) The operators that enable the on-the-fly domain specific adaptations.	Waterfall-like approach in domain-specific adaptations of modelling languages (or more in general terms meta-modelling).	Conceptual understanding of: Integration between modelling and meta-modelling. Operators for the on-the-fly domain-specific adaptations of modelling languages. Gained knowledge used as input for the prototype implementation.	Section 5
<i>Ontology-aided approach.</i>	An approach for the seamless alignment between the graphical and machine representation of modelling languages. It includes (1) the specification of the ontology architecture (2) semantic rules for the propagation of changes between the two knowledge representations.	Inconsistency between the graphical and machine interpretable representation of modelling languages caused by changes in the graphical modelling languages.	Conceptual understanding of: Ontology architecture for the specification of modelling languages: Palette Ontology, Modelling Language Ontology, and Domain Ontology Rules in support of operators for the propagation of changes from the graphical modelling language to the ontology. Gained knowledge used as input for the prototype implementation.	Section 5.3
<i>The AOAME Prototype.</i>	The prototype implements the agile and ontology-aided meta-modelling approach in a prototypical modelling environment. It automates the seamless alignment between the graphical and machine interpretable representation of modelling languages. Thus, consistency between the two representations is kept while the on-the-fly domain specific adaptations are performed.	Lack of automation for an approach that preserve consistency between the graphical and machine interpretable representation of modelling languages while on-the-fly domain-specific adaptations are performed.	Conceptual and practical understanding of: Dynamic generation and execution of rules. Functionalities that implement rules and support operators. System architecture. System implementation. Gained knowledge used to refine the main artefact.	Chapter 6

Table 65. Sub-artefacts contribution

#	Sub-Artefact Contribution	Description	Unsolved Problem	Gained Knowledge	Reference in the thesis
1	Model-based approach for extendable DSMLs.	A two-tier approach that allows the accommodation of new modelling requirements in a DSML. Domain-specific aspects are described declaratively in the form of an ontology.	Inability to keep up with the accommodation of new domain-specific aspects in application models.	Conceptual and practical understanding of: Meta-modelling, and its use for creating DSMLs Gained knowledge used as input for (2). Conceptual understanding of: Combination between models and ontologies. Gained knowledge used as input for (2) and (3).	Sub-section 4.1.4
2	DSML4PTM	A domain specific modelling-language for the patient transferal management. The model-based approach in (1) is further elaborated to create a DSML based on existing modelling languages.	Insufficient expressiveness of existing modelling languages to model domain-specific aspects of the patient transferal management.	Conceptual and practical understanding of: Domain-specific adaptations of modelling languages. Problems afflicting the domain-specific adaptations. Gained knowledge used as input for the agile meta-modelling approach (Table 64).	Section 4
3	Semantically-enhanced BPaaS	The domain-specific modelling language BPaaS (Business Process as a Service) is an extension of BPMN and allows modelling both business requirements and IT specifications of Cloud services.	Insufficient expressiveness of existing modelling languages to model business process requirements and IT specifications of cloud services.	Conceptual and practical understanding of: Domain-specific adaptations of a modelling language within BPaaS application domain. Problems afflicting the domain-specific adaptations. Gained knowledge used as input for the agile meta-modelling approach (see Table 64).	Section 4.2
	Semantically-enhanced BPaaS	The semantically-enhanced approach of BPaaS consists of an instantiation of the semantic lifting approach for the smart matchmaking between business requirements and IT specifications.	Difficulties to identify Cloud services or workflows by non-IT experts.	Conceptual and practical understanding of: Approaches that combine models with ontologies. Ontology conceptualisation and structure.	Section 4.2

			Semantic lifting (or annotation) and deriving drawbacks. Gained knowledge used as input for (4).		
4	Ontology-based meta-modelling approach.	This approach grounds the graphical modelling constructs with the ontology concepts, which reside in the meta-model.	Inconsistency between graphical models and ontology given by the separation of the two knowledge representations.	Conceptual understanding of: Ontology-based meta-modelling conceptualisation and structure. Gained knowledge used as input for (5) and for the ontology-aided meta-modelling approach (see Table 64).	Sub-section 5.3.1
5	Context-Adaptive Questionnaire.	This sub-artefact allows retrieving cloud services with the least number of questions. Whereas the questionnaire is specified in a business language, cloud services are specific in IT. Hence the artefact allows a Business-IT alignment using ontologies and rules. The artefact implements the ontology-based meta-modelling approach.	Inefficient specification of business process requirements and IT descriptions of cloud services.	Conceptual and practical understanding of: Ontologies and architecture specification for the ontology-based meta-modelling artefact. Rules and queries in SPARQL for the graphical models-ontology interaction, i.e. queries for the matchmaking and rules to infer knowledge. Mechanisms for the propagation of changes from the ontology to the graphical models. Gained knowledge used as input for an ontology-aided meta-modelling approach (see Table 64).	Sub-section 4.2.6.5.1

In design science research, artefacts and acquired knowledge should contribute to the practice and to the body of knowledge (Hevner and Chatterjee 2010). The following two sections describe the contributions made by this research.

8.2.2 Contribution to Practice

The acquired knowledge and understanding in this research work were communicated to practice by means of research projects and implementation of real-world application scenarios.

- (1) Contributions to modelling and domain experts in the patient transferal management: Modelling and domain experts that deal with patient transferal management benefit from the model-based approach used to create a Domain-Specific Modelling Language for Patient Transferal Management (DSML4PTM), which is implemented in ADOxx. This mainly refers to the way several existing modelling languages were conceptually extended and integrated with each other to accommodate as far as possible domain-specific requirements. Both the approach and the technical solution were communicated within the results of the Swiss research project Patient-Radar. Hence, the solution was made available to experts, which enabled use of the modelling tool in practice. This was validated in a workshop held at the Institute for Information and Process Management (IPM-FHSG) of the University of Applied Sciences St. Gallen with a total of five modelling experts and one domain expert. In the workshop experts worked on domain-specific models built with DSML4PTM and reflecting real-case scenarios. The workshop was
- (2) Contributions to modelling experts in Business Process as a Service (BPaaS) modelling experts that deal with the paradigm BPaaS can benefit from
 - (a) the BPMN extension that allows modelling Cloud service requirements and specifications;
 - (b) a semantic-enriched approach for the model annotation with ontologies as well as for the matchmaking between the business requirements and IT specifications of Cloud service;
 - (c) an ontology-based meta-modelling approach for the quick identification of Cloud services.

During the European research project CloudSocket each of the above contributions were integrated into prototypical products of the industry partner BOC⁶⁴.

- (3) Contribution to innovation experts: innovation experts can benefit from the AOAME approach to foster problem understanding and ideation in innovation processes. The AOAME approach was first communicated by myself to innovation experts from the Herman Hollerith Zentrum⁶⁵ (HHZ), during a visit in June 2017. The HHZ is an institute of the University of Reutlingen (in Germany) characterized by a high degree of synergy with local small and medium companies. AOAME was extended to integrate the pre-defined set of modelling elements of SAP Scenes, which are used in innovation processes to create storyboards. Hence, the AOAME approach went through an additional iteration of the Design Science Research (DSR) cycle, resulting in an extended artefact named AOAME4Scenes (Laurenzi et al., 2020). The latter allows adapting and extending the SAP Scenes to create storyboards of unforeseen scenarios. AOAME4Scenes is currently being evaluated in practice in a project with the company

⁶⁴ <https://ch.boc-group.com/>

⁶⁵ <https://www.hhz.de/home/>

aYo⁶⁶. In order to facilitate its use by practitioners AOAME4Scenes was deployed on the Cloud platform Heroku⁶⁷.

8.2.3 Contribution to the Body of Knowledge

The contribution of this research work to the body of knowledge spans the research fields of Domain-Specific Modelling Language, Meta-Modelling, Enterprise Modelling and Ontologies. The communication occurred primarily through publications in conference proceedings and in scientific books as well as through conference presentations.

The contribution to the body of knowledge is chronologically presented in the following list:

1. An early result of a model-based approach creates an extendable domain-specific modelling language that foresees the declarative description of domain-specific aspects in the form of an ontology (Laurenzi, 2014; Reimer & Laurenzi, 2014). The approach maps the language constructs with a machine-interpretable representation, to maintain the application models executable. The research activity contributed to raise awareness of approaches that create DSMLs, which declaratively describe domain-specific aspects as well as increasing awareness of approaches that combine models and ontologies. The approach was further elaborated in the two approaches described below in (2) and (3).
2. A domain-specific modelling language for the patient transferal management (DSML4PTM) (Laurenzi et al., 2017), which is then analysed to contribute to the elicitation of requirements for the new agile approach (Section 4)
3. A domain-specific modelling language for business process as a service (BPaaS DSML) (Hinkelmann, Laurenzi et al., 2016). The practice of engineering BPaaS was analysed to contribute to the elicitation of requirements for the new agile approach (Section 4.2).
4. An approach for the semantic lifting of graphical modelling languages and models within the context of BPaaS (Hinkelmann, Laurenzi, et al., 2016; Griesinger et al., 2017). The experience of semantic lifting increased awareness of the problems associated with it. Lessons learned were beneficial in devising the new ontology-based meta-modelling approach (see Sub-section 5.2.2). The ontology-based meta-modelling approach was proposed as an alternative to semantic lifting. The approach is published in (Hinkelmann, Laurenzi et al., 2018) and is described in Sub-section 5.3.1.
5. The ontology-based meta-modelling approach was applied to a context-adaptive and ontology-based questionnaire in BPaaS (Kritikos, Laurenzi et al., 2018) (see Sub-section 4.2.6.5.1).
6. An agile meta-modelling approach avoids sequential engineering phases following the publication of a critical analysis of the waterfall-type engineering lifecycle for the creation of DSMLs (Laurenzi, Hinkelmann, Izzo, et al., 2018) (see also Section 4.3). It fosters (1) agility in domain-specific adaptations of modelling languages (or more in general terms meta-modelling) and (2) promotes the tight cooperation between language engineers and domain experts. The approach integrates modelling with meta-modelling in the same modelling component. Such integration included a set of

⁶⁶ <https://www.ayo4u.com>

⁶⁷ the triplestore can be reached in the following link: <https://fusekiherokutest.herokuapp.com/dataset.html> and the GUI in the following link: <https://demo-angular-webapp.herokuapp.com/modeller>.

operators for the on-the-fly domain-specific adaptations and are published in (Laurenzi et al., 2018) as well as reported in Section 5.2.

7. The combination of ontology-aided and agile meta-modelling (Laurenzi, Hinkelmann, & van der Merwe, 2018). The approach was elaborated with an ontology architecture containing (1) the specification of an ontology-based meta-model (Sub-section 5.3.2) and (2) semantic rules in support of the operators (Sub-section 5.3.3). The approach was implemented in the agile and ontology-aided modelling environment (AOAME), which is described in Chapter 6.
8. The agile and ontology-aided meta-modelling approach was applied to create AOAME4Scenes (Laurenzi et al., 2020), a domain-specific modelling language for the creation of storyboards. The approach fosters problem understanding and ideation in innovation processes like Design Thinking. It has been shown that it has advantages compared to a non-agile approach (Miron et al., 2019) – see also Sub-section 7.4.7.

8.3 Methodological Suitability

This section describes the suitability of the methodology Design Science Research, chosen to conduct the research work. For this, the guidelines for design science in information systems research proposed by Hevner et al. (2004) were considered. The guidelines support evaluating the combination of this research work with design science research. In Table 66 research aspects of this thesis are described to fit the intent of each guideline. Sections also cover each mentioned research aspect.

Table 66. Adherence of the research to DSR guidelines from Hevner et al. (2004)

Guideline	Description	Research Aspects
<i>Design as an Artefact</i>	Design-science research must produce a viable artefact in the form of a construct, a model, a method, or an instantiation.	The artefact of this research is an agile and ontology-aided meta-modelling as a method (Chapter 5). The main contributions to this artefact are: - An agile meta-modelling approach as a method (Section 5). - An ontology-aided approach for the agile meta-modelling as a method (Section 5.3). - The prototype AOAME as implementation and instantiation (Chapter 6).
<i>Problem Relevance</i>	The objective of design-science research is to develop technology-based solutions to important and relevant business problems.	Challenges and requirements for agility are derived from two cases that tackle real-world problems as well as from expert interviews in industry (Chapter 4).
<i>Design Evaluation</i>	The utility, quality, and efficacy of a design artefact must be rigorously demonstrated via well-executed evaluation methods.	In order to rigorously evaluate the agile and ontology-aided meta-modelling approach an evaluation strategy was built and described in Section 7.2. The strategy was built according to relevant literature on evaluation of DSR artefacts. Among others, the strategy indicated the methods and evaluation criteria against which the artefact was evaluated. Hence, “Prototype” and “Illustrative Scenario” were chosen as most appropriate method types. Real-world use cases (i.e. illustrative scenarios) were implemented through the prototype to evaluate the operationability (also called utility in more general terms) and generality of the approach (Section 7.4). The correct design of the artefact was evaluated by demonstrating how the prototype satisfies the design requirements (Section 7.3).
<i>Research Contributions</i>	Effective design-science research must provide clear and verifiable contributions in the areas of the design artefact, design foundations, and/or design methodologies.	The proposed agile and ontology-aided meta-modelling approach was presented in a reproducible way (Chapter 5) and it contributed to several scientific publications (see Sub-section 8.2.3).

<i>Research Rigor</i>	Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artefact.	This research work follows design-science research. To increase understanding of the problem, the case study strategy is also employed. Namely, two cases were presented in which DSMLs were created following the AMME Lifecycle (see Sections 4 and 4.2). A qualitative analysis was then performed over the two cases and problems were revealed (Section 4.3). Interviews were conducted (Section 4.4) and their analysis is described in (Sub-section 4.4.3). For the creation of semantic rules (Sub-section 5.3.3) the Grüninger and Fox (1995) methodology was adopted. Finally, two method types “Prototype” and “Illustrative Scenario” were considered for the evaluation. Illustrative scenarios reflect the real world to demonstrate relevance in the practice.
<i>Design as a Search Process</i>	The search for an effective artefact requires utilizing available means to reach desired ends while satisfying laws in the problem environment	The artefact is designed through iterative cycles. The implemented artefact was iteratively tested against the evaluation criteria (Sub-sections 7.3 and 7.4). Hevner et al. (2004) refer to it as the Generate/Test Cycle, which aims to discover an effective solution to a problem. As stressed in Section 7.5, such a cycle was helpful to identify deficiencies of the prototype, which were addressed to refine it.
<i>Communication of Research</i>	Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.	The results of this research were disseminated in peer-reviewed conference proceedings as well as presented to project application partners and entities external to projects like innovation experts (see Sub-section 8.2.1).

8.5 Concluding Summary

This research work starts by stressing the lack of agility in current engineering processes for the development of DSMLs. This problem has been identified from both the literature investigation and the analysis of two case studies. The two case studies describe the development of two DSMLs, which address two different application domains, the Patient Transferal Management and the Business Process as a Service. Findings revealed that agility is hindered by the sequential engineering lifecycle, where modelling and evaluation activities cannot start before the DSML is deployed for usage. This is problematic as it tends to keep the two key roles for DSML engineering - the language engineer and the domain expert – separate. The side effect is that extra engineering iterations are likely to be needed until a satisfactory version of a DSML is achieved. Each engineering iteration, however, is time-consuming as a new requirement is first captured and documented, then conceptualised, implemented and finally a new version of a DSML is deployed so that modelling and evaluation can re-occur.

To address these problems an agile meta-modelling approach has been conceived. The approach integrates language engineering and modelling into one single component. In contrast to current meta-modelling approaches, language engineering, modelling and evaluation activities can be quickly interleaved. The quick interleave is enabled by on-the-fly domain-specific adaptations, which are supported by a set of ten operators. As a result, the agile meta-modelling approach fosters the tight cooperation between the language engineers and domain experts from an early stage and avoids the sequential engineering phases. Thus, the creation of the quick and high quality of DSMLs is promoted.

It was shown, however, that the use of the operators can lead to some drawbacks. In particular, the meaning of the newly created modelling constructs can be misunderstood as they are left to the human interpretation only. To overcome this drawback the agile meta-modelling was supplemented with an ontology-aided approach. The latter embeds the specifications of modelling languages into an ontology. The meaning of the modelling constructs is, therefore, made not only explicit but also machine-interpretable. Models that are built with such a ontology-based modelling language are also ontology-based, thus leading to increase their shared understanding as well as they are machine interpretable. In order to preserve the machine interpretability of modelling languages, eleven semantic rules were created. The rules support the afore-mentioned ten operators for the propagation of language adaptations from the graphical to the machine-interpretable representation.

The agile and ontology-aided meta-modelling approach was developed in the modelling environment AOAME. The latter enables the automated preservation of consistency between graphical and machine-interpretable knowledge when domain-specific adaptations occur. AOAME was then used to evaluate the approach with respect to the correct design, operationability and generality. The approach has been successfully applied in three different application domains, the Patient Transferal Management, Business Process as a Service and Innovation Processes. The scientific contribution spans the research fields of Domain-Specific Modelling Language, Meta-Modelling, Enterprise Modelling and Ontologies. Finally, three different future research have been pointed, which build upon the presented approach. Namely, the investigation of approaches that deal with the inconsistencies that operators can cause to existing models; the injection of constraints in the ontology-based meta-model; and the exploitation of the ontologies for an intelligent modelling assistance.

BIBLIOGRAPHY

- Abecker, A., Bernardi, A., Hinkelmann, K., Kühn, O., & Sintek, M. (1998). Toward a Technology for Organizational Memories. *IEEE Intelligent Systems and Their Applications*, 13(3), 40–48.
- Agar, M. (1980). *The Professional Stranger: An Informal Introduction to Ethnography*. New York: Nueva York EUA Academic Press.
- Aier, S., & Fischer, C. (2011). Criteria of Progress for Information Systems Design Theories. *Information Systems and E-Business Management*, 9(1), 133–172. <https://doi.org/10.1007/s10257-010-0130-8>
- Allemang, D., & Hendler, J. A. (2011). *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL (2nd ed.)*. San Francisco: Morgan Kaufmann/Elsevier.
- Amziani, M., Melliti, T., & Tata, S. (2012). A Generic Framework for Service-Based Business Process Elasticity in the Cloud. In *Proceedings of the 10th International Conference on Business Process Management*, 194–199. Springer-Verlag. https://doi.org/10.1007/978-3-642-32885-5_15
- APQC. (2014). Process Classification Framework Version 6.1.1. <https://www.apqc.org/resource-library/resource-listing/apqc-process-classification-framework-pcf-cross-industry-pdf-5>
- Aquino, N., Vanderdonckt, J., Panach, J. I., & Pastor, O. (2011). Conceptual Modelling of Interaction. *Handbook of Conceptual Modeling*, 335–358. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-15865-0_10
- Aßmann, U., Zschaler, S., & Wagner, G. (2006). Ontologies, Meta-Models, and the Model-Driven Paradigm. In C. Calero, F. Ruiz, & M. Piattini (Eds.), *Ontologies for Software Engineering and Software Technology*, 249–273. Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-34518-3_9
- Atkinson, C., & Kuhne, T. (2003). Model-Driven Development: A Metamodeling Foundation. *IEEE Software*, 20(5), 36–41. <https://doi.org/10.1109/MS.2003.1231149>
- Atkinson, Colin, Gerbig, R., & Fritzsche, M. (2013). Modeling Language Extension in the Enterprise Systems Domain. In *Proceedings of the 17th IEEE International Enterprise Distributed Object Computing Conference*, 49–58. <https://doi.org/10.1109/EDOC.2013.15>
- Atkinson, Colin, Gerbig, R., & Fritzsche, M. (2015). A Multi-Level Approach to Modeling Language Extension in the Enterprise Systems Domain. *Information Systems*, 54, 289–307. <https://doi.org/10.1016/j.is.2015.01.003>
- Atkinson, Colin, & Kühne, T. (2008). Reducing Accidental Complexity in Domain Models. *Software & Systems Modeling*, 7(3), 345–359. Springer Berlin Heidelberg. <https://doi.org/10.1007/s10270-007-0061-0>
- Atkinson, Colin, & Kühne, T. (2017). On Evaluating Multi-Level Modeling. In *Workshop Proceedings of MODELS 2017 Satellite Event*, 274–277. CEUR-WS.org. https://www.wi-inf.uni-duisburg-essen.de/MULTI2017/wp-content/uploads/2017/09/Atkinson-Kuehne_2017_On-Evaluating-Multi-Level-Modeling-1.pdf
- Auer, S., Berners-Lee, T., Bizer, C., & Heath, T. (2015). LDOW 2013: The 8th Workshop on Linked Data on the Web. In *Proceedings of the 24th International Conference on World Wide Web - WWW '15 Companion*, 1549–1550. New York, USA: ACM Press. <https://doi.org/10.1145/2740908.2742577>
- Avazpour, I., Grundy, J., & Grunske, L. (2015). Specifying Model Transformations by Direct Manipulation using Concrete Visual Notations and Interactive Recommendations. *Journal of Visual Languages & Computing*, 28, 195–211. <https://doi.org/10.1016/j.jvlc.2015.02.005>
- Azzini, A., Braghin, C., Damiani, E., & Zavatarelli, F. (2013). Using Semantic Lifting for improving

- Process Mining: A Data Loss Prevention System Case Study. In *SIMPDA 2013: 3rd International Symposium on Data-driven Process Discovery and Analysis*, 62-73. CEURWS.org.
- Baader, F. (2011). What's New in Description Logics. In *Informatik-Spektrum*, 34(5), 434-442. Springer-Verlag. <https://doi.org/10.1007/s00287-011-0534-y>
- Baader, F., & Nutt, W. (2003). *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press. <https://dl.acm.org/citation.cfm?id=885749>
- Baker, T., & Collier, D. A. (2005). The Economic Payout Model for Service Guarantees. *Decision Sciences*, 36(2), 197–220. Wiley Online Library. <https://doi.org/10.1111/j.1540-5414.2005.00071.x>
- Barišić, A., Amaral, V., & Goulão, M. (2018). Usability Driven DSL Development with USE-ME. *Computer Languages, Systems & Structures*, 51, 118–157. North-Holland, Elsevier. <https://doi.org/10.1016/j.cl.2017.06.005>
- Battistutti, O. C., & Bork, D. (2017). *Tacit to Explicit Knowledge Conversion*. Cognitive Processing, 18, 461–477. SpringerLink. <https://doi.org/10.1007/s10339-017-0825-6>
- Beck, K., & Kent. (2000). *Extreme Programming eXplained: Embrace Change*. Addison-Wesley. <https://dl.acm.org/citation.cfm?id=318762>
- Becker, J. (2014). Interview with Reinhard Schütte on “Managing Large-Scale BPM Projects”. *Business & Information Systems Engineering*, 6(4). Springer. <http://aisel.aisnet.org/bise/vol6/iss4/6>
- Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Highsmith, J., Thomas, D. (2001). *Manifesto for Agile Software Development*. <http://agilemanifesto.org/>
- Benyon, D., Bental, D., & Green, T. (1999). *Conceptual Modeling for User Interface Development*. Springer-Verlag.
- Berardi, D., Calvanese, D., & De Giacomo, G. (2005). Reasoning on UML class diagrams. *Artificial Intelligence*, 168(1–2), 70–118. <https://doi.org/10.1016/J.ARTINT.2005.05.003>
- Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web: A New Form of Web Content that is Meaningful to Computers will Unleash a Revolution of New Possibilities. *Scientific American*, 284 (5), 34-43.
- Bézivin, J. (2005). On the Unification Power of Models. *Software & Systems Modeling*, 4(2), 171–188. <https://doi.org/10.1007/s10270-005-0079-0>
- Bohdana, S., Karwowski, W., & Layer, J. K. (2007). A Review of Enterprise Agility: Concepts, Frameworks, and Attributes. *International Journal of Industrial Ergonomics*, 37(5), 445–460. <https://doi.org/10.1016/J.ERGON.2007.01.007>
- Bork, Dominik, & Fill, H.-G. (2014). Formal Aspects of Enterprise Modeling Methods: A Comparison Framework. In *Proceedings of the 47th Hawaii International Conference on System Sciences*, 3400–3409. IEEE. <https://doi.org/10.1109/HICSS.2014.422>
- Bork, Dominik, & Alter, S. (2018). Relaxing Modeling Criteria to Produce Genuinely Flexible, Controllable, and Usable Enterprise Modeling Approaches. In *Proceedings of the 9th International Workshop on Enterprise Modeling and Information Systems Architectures*, 46–50. Rostock, Germany. <http://eprints.cs.univie.ac.at/5575/>
- Bork, Dominik, Buchman, R. A., Karagiannis, D., Lee, M., & Miron, E.-T. (2019). An Open Platform for Modeling Method Conceptualization: The OMiLAB Digital Ecosystem. *Communications of the Association for Information Systems*, 44. <https://doi.org/10.17705/1CAIS.04432>
- Brachman, R. J., Levesque, H. J., & Pagnucco, M. (2004). *Knowledge representation and reasoning*. Morgan Kaufmann Publishers Inc. <http://dl.acm.org/citation.cfm?id=2821570>
- Bräuer, M., & Lochmann, H. (2008). An Ontology for Software Models and Its Practical Implications for Semantic Web Reasoning. *The Semantic Web: Research and Applications*, 34–48. Berlin,

- Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-68234-9_6
- Braun, R. (2015a). BPMN Extension Profiles -- Adapting the Profile Mechanism for Integrated BPMN Extensibility. In *Proceedings of the 17th Conference on Business Informatics*, 133–142. IEEE. <https://doi.org/10.1109/CBI.2015.41>
- Braun, R. (2015b). Towards the State of the Art of Extending Enterprise Modeling Languages. In *Proceedings of the 3rd International Conference on Model-Driven Engineering and Software Development*. Angers, France: IEEE. <https://doi.org/978-989-758-136-6>
- Braun, R. (2016). *Extensibility of Enterprise Modelling Languages*. Technischen Universität Dresden. <http://www.qucosa.de/fileadmin/data/qucosa/documents/21987/phd-thesis.pdf>
- Braun, R., & Esswein, W. (2014). Classification of Domain-Specific BPMN Extensions. *The Practice of Enterprise Modeling*, 42–57. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-45501-2_4
- Braun, R., Schlieter, H., Burwitz, M., & Esswein, W. (2015). Extending a Business Process Modeling Language for Domain-Specific Adaptation in Healthcare. In *Wirtschaftsinformatik Proceedings*, 32. Association for Information Systems. <https://aisel.aisnet.org/wi2015/32>
- Braun, R., Schlieter, H., Burwitz, M., & Esswein, W. (2016). BPMN4CP Revised – Extending BPMN for Multi-Perspective Modeling of Clinical Pathways. In *Proceedings of the 49th Hawaii International Conference on System Sciences (HICSS)*, 3249–3258. IEEE. <https://doi.org/10.1109/HICSS.2016.407>
- Bridgeland, D. M., & Zahavi, R. (2009). *Business Modeling: A Practical Guide to Realizing Business Value*. Morgan Kaufmann/Elsevier. https://books.google.ch/books/about/Business_Modeling.html?id=AzeN11NEoNAC&printsec=frontcover&source=kp_read_button&redir_esc=y#v=onepage&q&f=false
- Bryman, A. (2006). Integrating Quantitative and Qualitative Research: How is it Done? *Qualitative Research*, 6(1), 97–113. SAGE Journals. <https://doi.org/10.1177/1468794106058877>
- Buchmann, R. A. (2016). Modeling Product-Service Systems for the Internet of Things: The ComVantage Method. In *Domain-Specific Conceptual Modeling*, 417–437. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-39417-6_19
- Burlton, R. T., Ross, R. G., & Zachman, J. A. (2017). *The Business Agility Manifesto Building for Change*. https://busagilitymanifesto.org/images/pdfs/Business_Agility_Manifesto.pdf
- Burwitz, M., Schlieter, H., & Esswein, W. (2013). Modeling Clinical Pathways - Design and Application of a Domain-Specific Modeling Language. *Wirtschaftsinformatik Proceedings 2013*. <http://aisel.aisnet.org/wi2013/83>
- Cadavid, J. J., Quintero, J. B., Lopez, D. E., & Hincapié, J. A. (2009). A Domain Specific Language to Generate Web Applications. In Antonio B., João A., & Raquel A. (Eds.), *Memorias de la XII Conferencia Iberoamericana de Software Engineering (CIBSE 2009), Medellin, Colombia, Abril 13-17, 2009*, 139–144.
- Cardoso, Y. C. (2010). *Creation and Extension of Ontologies for Describing Communications in the Context of Organizations*. Universidade Nova de Lisboa.
- Ceh, I., Crepinsek, M., Kosar, T., & Mernik, M. (2011). Ontology driven development of domain-specific languages. *Computer Science and Information Systems*, 8(2), 317–342. <https://doi.org/10.2298/CSIS101231019C>
- Chavula, C., & Maria Keet, C. (2015). An Orchestration Framework for Linguistic Task Ontologies. In G. P. Garoufallou E., Hartley R. (Ed.), *Metadata and Semantics Research. MTSR 2015. Communications in Computer and Information Science*. 544th ed., 3–14. Springer, Cham. https://doi.org/10.1007/978-3-319-24129-6_1
- Chen, W., & Hirschheim, R. (2004). A Paradigmatic and Methodological Examination of Information Systems research from 1991 to 2001. *Information Systems Journal*, 14(3), 197–235.

<https://doi.org/10.1111/j.1365-2575.2004.00173.x>

- Chiprianov, V., Kermarrec, Y., & Rouvrais, S. (2012). Extending Enterprise Architecture Modeling Languages. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing - SAC '12* (p. 1661). New York, New York, USA: ACM Press.
<https://doi.org/10.1145/2245276.2232044>
- Chiprianov, V., Kermarrec, Y., Rouvrais, S., & Simonin, J. (2013). Extending Enterprise Architecture Modeling Languages for Domain Specificity and Collaboration: Application to Telecommunications Service Design. *Software & Systems Modeling*, 13, 963-974. Springer.
<https://doi.org/10.1007/s10270-012-0298-0>
- Cho, H., Gray, J., & Syriani, E. (2012). Creating visual Domain-Specific Modeling Languages from end-user demonstration. In *Proceedings of the 4th International Workshop on Modeling in Software Engineering (MISE)*, 22–28. IEEE. <https://doi.org/10.1109/MISE.2012.6226010>
- Clark, T., van den Brand, M., Combemale, B., & Rumpe, B. (2015). Conceptual Model of the Globalization for Domain-Specific Languages (pp. 7–20). Springer, Cham.
https://doi.org/10.1007/978-3-319-26172-0_2
- Cleven, A., Gubler, P., & Hüner, K. M. (2009). Design alternatives for the evaluation of design science research artifacts. In *Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology - DESRIST '09* (p. 1). New York, New York, USA: ACM Press. <https://doi.org/10.1145/1555619.1555645>
- Cognini, R., Corradini, F., Polini, A., & Re, B. (2016). Business Process Feature Model: An Approach to Deal with Variability of Business Processes. In *Domain-Specific Conceptual Modeling*, 171–194. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-39417-6_8
- Creswell, J. (2002). *Research Design: Quantitative and Qualitative Approaches* (2nd ed.). Thousand and Oaks: CA: Sage.
- Cronje, J. (2011). The ABC (Aim, Belief, Concern) Instant Research Question Generator. Unpublished Manuscript.
<https://pdfs.semanticscholar.org/e132/2ddf75705095fd33dc573f489861c2aaf11f.pdf>
- Crotty, M. (1998). *The Foundations of Social Research: Meaning and Perspective in the Research Process*. Sage Publications.
- De Angelis, G., Pierantonio, A., Polini, A., Re, B., Thönssen, B., & Woitsch, R. (2016). Modeling for Learning in Public Administrations—The Learn PAd Approach. In *Domain-Specific Conceptual Modeling*, 575–594. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-39417-6_26
- De Bruijn, J. (2003). *Using Ontologies - Enabling Knowledge Sharing and Reuse on the Semantic Web. October*. Galway, Ireland.
- De Leenheer, P., & Mens, T. (2008). Ontology Evolution. In Martin Hepp, P. De Leenheer, A. De Moor, & Y. Sure (Eds.), *Ontology Management - Semantic Web, Semantic Web Services, and Business Applications*, 131–176. SpringerScience + Business Media Inc.
- Decker, G., Kopp, O., Leymann, F., & Weske, M. (2007). BPEL4Chor: Extending BPEL for Modeling Choreographies. In *IEEE International Conference on Web Services (ICWS 2007)*, 296–303. IEEE. <https://doi.org/10.1109/ICWS.2007.59>
- Delfmann, P., Herwig, S., Lis, Ł., Stein, A., Tent, K., & Becker, J. (2010). Pattern Specification and Matching in Conceptual Models - A Generic Approach Based on Set Operations. *Enterprise Modelling and Information Systems Architectures (EMISAJ)*, 5(3), 24–43.
<https://doi.org/10.18417/emisa.5.3.2>
- Deloitte. (2019). *Wisdom of Enterprise Knowledge Graphs*.
<https://www2.deloitte.com/content/dam/Deloitte/de/Documents/operations/knowledge-graphs-pov.pdf>

- Den Haan, J. (2009). *An Enterprise Ontology based approach to Model-Driven Engineering*. TU Delft, Delft University of Technology. <http://repository.tudelft.nl/view/ir/uuid:e2093132-9db7-4cba-bc68-9355f93cb9e3/>
- Deng, Q., & Ji, S. (2018). A Review of Design Science Research in Information Systems: Concept, Process, Outcome, and Evaluation. *Pacific Asia Journal of the Association for Information Systems*, 10(1), 1–36. <https://doi.org/10.17705/1pais.10101>
- Didonet Del Fabro, M., & Valduriez, P. (2009). Towards the efficient development of model transformations using model weaving and matching transformations. *Software & Systems Modeling*, 8(3), 305–324. <https://doi.org/10.1007/s10270-008-0094-z>
- Dietz, J. L. G. (2006). *Enterprise Ontology: Theory and Methodology*. Information Systems & Applications. Springer-Verlag Berlin Heidelberg. <https://link.springer.com/book/10.1007%2F3-540-33149-2>
- Dove, R. (1999). Knowledge Management, Response Ability, and the Agile Enterprise. *Journal of Knowledge Management*, 3(1), 18–35. <https://doi.org/10.1108/13673279910259367>
- Drucker, P. F. (2001). The Next Society. *The Economist*. <https://www.economist.com/node/770819>
- Dumas, M., La Rosa, M., Mendling, J., & Reijers, H. (2018). *Fundamentals of Business Process Management* (2nd ed.). Springer-Verlag Berlin Heidelberg. <https://doi.org/10.1007/978-3-662-56509-4>
- Easterby-Smith, M., Thorpe, R., Jackson, P., & Lowe, A. (2008). *Management Research* (3rd ed.). London: Sage.
- EC Cloud Select Industry Group (C-SIG). (2014). Cloud Service Level Agreement Standardization Guidelines. <https://ec.europa.eu/digital-single-market/en/news/cloud-service-level-agreement-standardisation-guidelines>
- Efendioglu, N., Woitsch, R., Utz, W., & Falcioni, D. (2017). ADOxx Modelling Method Conceptualization Environment. *ASTESJ* (2), 3, 125-136. <https://astesj.com/v02/i03/p17/>
- Ellis, T. J., & Levy, Y. (2008). Framework of Problem-based Research: A Guide for Novice Researchers on the Development of a Research-Worthy Problem. *Informing Science: International Journal of an Emerging Transdiscipline*, 11, 17–33.
- Emmenegger, S., Hinkelmann, K., Laurenzi, E., Thönssen, B., Witschel, H. F., & Zhang, C. (2016). Workplace learning-Providing Recommendations of Experts and Learning Resources in a Context-Sensitive and Personalized Manner: An Approach for Ontology Supported Workplace Learning. In *MODELSWARD 2016 - Special Session on Learning Modeling in Complex Organizations. Proceedings of the 4th International Conference on Model-Driven Engineering and Software Development*. Rome.
- Emmenegger, Sandro, Hinkelmann, K., Laurenzi, E., Martin, A., Thönssen, B., Witschel, H. F., & Zhang, C. (2017). An Ontology-Based and Case-Based Reasoning Supported Workplace Learning Approach. In *Communications in Computer and Information Science*, 333–354. Springer, Cham. https://doi.org/10.1007/978-3-319-66302-9_17
- Emmenegger, Sandro, Hinkelmann, K., Laurenzi, E., & Thönssen, B. (2013). Towards a Procedure for Assessing Supply Chain Risks Using Semantic Technologies. In A. Fred, J., Dietz, K. Liu, & J. Filipe (Eds.), *Knowledge Discovery, Knowledge Engineering and Knowledge Management*, 415, 393–409. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-54105-6_26
- Eurostat. (2017). *Healthcare expenditure statistics*. http://ec.europa.eu/eurostat/statistics-explained/index.php/Healthcare_expenditure_statistics
- Fanesi, D., Cacciagrano, D. R., & Hinkelmann, K. (2015). Semantic Business Process Representation to Enhance the Degree of BPM Mechanization - An Ontology. In *2015 International Conference on Enterprise Systems (ES)*, 21–32. IEEE. <https://doi.org/10.1109/ES.2015.10>
- Favre, J. M. (2005). Foundations of Meta-Pyramids: Languages vs. Metamodels - Episode II: Story of

- Thotus the Baboon. In Jean Bezin and Reiko Heckel (Ed.), *Language Engineering for Model-Driven Software Development*. Dagstuhl, Germany: Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.
<http://drops.dagstuhl.de/opus/volltexte/2005/21>
- Felfernig, A., Friedrich, G. E., & Jannach, D. (2000). UML as Domain Specific Language for the construction of Knowledge-Based Configuration System. *International Journal of Software Engineering and Knowledge Engineering*, 10(04), 449–469.
<https://doi.org/10.1142/S0218194000000249>
- Fill, H.-G. (2011). On the Conceptualization of a Modeling Language for Semantic Model Annotations. In C. Salinesi & O. Pastor (Eds.), *Advanced Information Systems Engineering Workshops*, 83, 134–148. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-22056-2_14
- Fill, H.-G., Schremser, D., & Karagiannis, D. (2013). A Generic Approach for the Semantic Annotation of Conceptual Models Using a Service-Oriented Architecture. *International Journal of Knowledge Management*, 9(1), 76–88. <https://doi.org/10.4018/jkm.2013010105>
- Fill, H. G., & Karagiannis, D. (2013). On the Conceptualisation of Modelling Methods Using the ADOxx Meta Modelling Platform. *Enterprise Modelling and Information Systems Architectures - An International Journal*, 8(1). <http://eprints.cs.univie.ac.at/3657/>
- Fill, H. G., Redmond, T., & Karagiannis, D. (2012). FDMM: A Formalism for Describing ADOxx Meta Models and Models. In Leszek A. Maciaszek, Alfredo Cuzzocrea, & José Cordeiro (Eds.), *ICEIS 2012 - Proceedings of the 14th International Conference on Enterprise Information Systems, Volume 3, Wroclaw, Poland, 28 June - 1 July, 2012*, 133–144. SciTePress.
- Fowler, M. (2011). *Domain-specific languages*. Upper Saddle River: Addison-Wesley.
- Fox, M. S. (1992). The TOVE project towards a common-sense model of the enterprise. In *Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, 25–34. Berlin/Heidelberg: Springer-Verlag. <https://doi.org/10.1007/BFb0024952>
- Fox, M. S., & Gruninger, M. (1998). Enterprise Modeling. *AI Magazine*, 19(3), 109.
<https://doi.org/10.1609/AIMAG.V19I3.1399>
- France, R. B., Ghosh, S., Dinh-Trong, T., & Solberg, A. (2006). Model-Driven Development Using UML 2.0: Promises and Pitfalls. *Computer*, 39(2), 59–66. <https://doi.org/10.1109/MC.2006.65>
- Frank, U. (2008). The MEMO Meta Modelling Language (MML) and Language Architecture.
<http://ideas.repec.org/p/zbw/udeicb/24.html>
- Frank, U. (2010). Outline of a Method for Designing Domain-Specific Modelling Languages. University of Duisburg Essen: ICB. <http://hdl.handle.net/10419/58163>
- Frank, U. (2011). *The MEMO Meta Modelling Language (MML) and Language Architecture. 2nd Edition*. ICB-Research Report No. 43. <https://www.econstor.eu/handle/10419/58154>
- Frank, U. (2013a). Domain-Specific Modeling Languages: Requirements Analysis and Design Guidelines. In *Domain Engineering* (pp. 133–157). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-36654-3_6
- Frank, U. (2013b). Multilevel Modeling Toward a New Paradigm of Conceptual Modeling and Information Systems Design. <https://doi.org/10.1007/s11576-014-0438-y>
- Frank, U. (2014a). Enterprise Modelling: The Next Steps. *Enterprise Modelling and Information Systems Architectures*, 9(1), 22–37. <https://doi.org/10.18417/EMISA.9.1.2>
- Frank, U. (2014b). Multi-Perspective Enterprise Modeling: Foundational Concepts, Prospects and Future Research Challenges. *Software & Systems Modeling*, 13(3), 941–962.
<https://doi.org/10.1007/s10270-012-0273-9>
- Frantz, R. Z., Quintero, A. M. R., & Corchuelo, R. (2011). A Domain-Specific Language to Design Enterprise Application Integration Solutions. *Int. J. Cooperative Inf. Syst.*, 143–176.

- Freund, J., & Rücker, B. (2016). *Real-life BPMN: using BPNM, CMMN and DMN to analyze, improve, and automate processes in your company* (3rd ed.).
- Gabriel, P., Goulão, M., & Amaral, V. (2011). Do Software Languages Engineers Evaluate their Languages? In *XIII Congresso Iberoamericano en 'Software Engineering' (CibSE'2010)*. <http://arxiv.org/abs/1109.6794>
- Gailly, F., Alkhaldi, N., Casteleyn, S., & Verbeke, W. (2017). Recommendation-Based Conceptual Modeling and Ontology Evolution Framework (CMOE+). *Business & Information Systems Engineering*, 59(4), 235–250. <https://doi.org/10.1007/s12599-017-0488-y>
- Gall, M. D., Gall, J. P., & Borg, W. (2006). *Educational Research: An Introduction* (8th ed.). New York: Longman.
- Gartner. (2014). *Taming the Digital Dragon: The 2014 CIO Agenda*.
- Gašević, D., Djuric, D., & Devedžić, V. (2009). Knowledge Representation. In *Model Driven Engineering and Ontology Development* (pp. 3–43). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-00282-3_1
- Giachetti, R. E. (2010). *Design of Enterprise Systems: Theory, Architecture, and Methods*. CRC Press. <https://www.crcpress.com/Design-of-Enterprise-Systems-Theory-Architecture-and-Methods/Giachetti/p/book/9781439818237>
- Giovanoli, C. (2019). Cloud Service Quality Model: A Cloud Service Quality Model-based on Customer and Provider Perceptions for Cloud Service Mediation. In *Proceedings of the 9th International Conference on Cloud Computing and Services Science (CLOSER 2019)* (pp. 241–248). SCITEPRESS. <https://doi.org/10.5220/0007587502410248>
- Glaser, B. G., & Strauss, A. (1967). *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Aldine Publishing.
- Gli Ospedali Svizzeri. (2017). H+. Retrieved 11 March 2017, from <http://www.hplus.ch/>
- Goldkuhl, G. (2011). Pragmatism vs Interpretivism in Qualitative Information Systems Research. *European Journal of Information Systems*, 21(2), 135–146. <https://doi.org/10.1057/ejis.2011.54>
- Goles, T., & Hirschheim, R. (2000). The Paradigm is Dead, the Paradigm is Dead...Long Live the Paradigm: the Legacy of Burrell and Morgan. *Omega*, 28(3), 249–268. <http://ideas.repec.org/a/eee/jomega/v28y2000i3p249-268.html>
- Gomez-Perez, A., Fernandez-Lopez, M., & Corcho, O. (2004). *Ontological Engineering* (4th ed.). Springer-Verlag London, UK.
- Götzinger, D., Miron, E.-T., & Staffel, F. (2016). OMiLAB: An Open Collaborative Environment for Modeling Method Engineering. In *Domain-Specific Conceptual Modeling*, 55–76. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-39417-6_3
- Gray, J., Fisher, K., Consel, C., Karsai, G., Mernik, M., & Tolvanen, J.-P. (2008). DSLs: The Good, the Bad, and the Ugly. In *Conference on Object Oriented Programming Systems Languages and Applications archive*. Nashville, United States: ACM. <http://hal.inria.fr/inria-00402566>
- Green, T. R. G., & Petre, M. (2008). Usability Analysis of Visual Programming Environments: a 'cognitive dimensions' framework. *Journal of Visual Languages and Computing*, 5(2), 1–17. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.35.815&rep=rep1&type=pdf>
- Gregg, D. G., Kulkarni, U. R., & Vinzé, A. S. (2001). Understanding the Philosophical Underpinnings of Software Engineering Research in Information Systems. *Information Systems Frontiers*, 3(2), 169–183. <https://doi.org/10.1023/A:1011491322406>
- Griesinger, F., Seybold, D., Wesner, S., Domaschka, J., Woitsch, R., Kritikos, K., ... Tuguran, C. V. (2017). BPaaS in Multi-cloud Environments - The CloudSocket Approach. In *European Space Projects: Developments, Implementations and Impacts in a Changing World*, 50–74. SCITEPRESS - Science and Technology Publications. <https://doi.org/10.5220/0007901700500074>

- Gronback, R. C. (2009). *Eclipse Modeling Project: A Domain-Specific Language (DSL) Toolkit* (1st ed.). Addison-Wesley Professional.
- Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2), 199–220. <https://doi.org/10.1006/knac.1993.1008>
- Grüninger, M., & Fox, M. S. (1995). Methodology for the Design and Evaluation of Ontologies. In *Proceedings of the 1995 International Joint Conference on AI*, 95, 1–10.
- Guarino, N., Guarino, N., & Giaretta, P. (1995). Ontologies and Knowledge Bases: Towards a Terminological Clarification. *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing*, 25-32. IOS Press.
<http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.320.8006>
- Guizzardi, G. (2007). On Ontology, Ontologies, Conceptualizations, Modeling Languages, and (Meta)Models. In *Conference on Databases and Information Systems IV*, 18–39. IOS Press Amsterdam. <https://doi.org/978-1-58603-715-4>
- Guizzardi, G. (2012). Ontological Foundations for Conceptual Modeling with Applications. In Ralyté J., Franch X., Brinkkemper S., Wrycza S. (eds) *Advanced Information Systems Engineering. CAiSE 2012. Lecture Notes in Computer Science*, vol 7328. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-31095-9_45
- Gulden, J., & Frank, U. (2010). MEMOCenterNG - A Full-Featured Modeling Environment for Organization Modeling and Model-Driven Software Development. In *Proceedings of the 22nd International Conference on Advanced Information Systems Engineering (CAiSE'10)*. Hammamet. <http://www.wi-inf.uni-duisburg-essen.de/FGFrank/>
- Haack, S. (1976). The Pragmatist Theory of Truth. *British Journal of Philosophical Science*, 27, 231–249. <http://philpapers.org/rec/HAATPT>
- Harel, D., & Rumpe, B. (2000). *Modeling Languages: Syntax, Semantics and All That Stuff, Part I: The Basic Stuff*. Weizmann Science Press of Israel. <https://dl.acm.org/citation.cfm?id=903627>
- Harel, D., & Rumpe, B. (2004). Meaningful Modeling: What's the Semantics of "Semantics"? *Computer*, 37(10), 64–72. IEEE Computer Society Press, Washington DC, United States. <https://doi.org/10.1109/MC.2004.172>
- Healey, M. J., & Rawlinson, M. B. (1994). Interviewing Techniques in Business and Management Research. In V. J. Wass & P. E. Wells (Eds.), *Principles and Practice in Business and Management Research*. Dartmouth: Aldershot.
- Hepp, M., Leymann, F., Domingue, J., Wahler, A., & Fensel, D. (2005). Semantic Business Process Management: a Vision towards using Semantic Web Services for Business Process Management. In *IEEE International Conference on e-Business Engineering (ICEBE'05)*, 535–540. IEEE. <https://doi.org/10.1109/ICEBE.2005.110>
- Heß, M., Kaczmarek, M., Frank, U., Podleska, L., & Täger, G. (2015). A Domain-Specific Modelling Language for Clinical Pathways in the Realm of Multi-Perspective Hospital Modelling. *ECIS 2015 Completed Research Papers*. <https://doi.org/10.18151/7217355>
- Hevner, A. (2007). A Three-Cycle View of Design Science Research. *Scandinavian Journal of Information Systems*, 19(2), 87–92.
<https://aisel.aisnet.org/cgi/viewcontent.cgi?article=1017&context=sjis>
- Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design Science in Information Systems Research. *MIS Q*, 28(1), 75–105. <http://dl.acm.org/citation.cfm?id=2017212.2017217>
- Hevner, Alan, & Chatterjee, S. (2010). *Design Research in Information Systems* (Vol. 22). Boston, MA: Springer US. <https://doi.org/10.1007/978-1-4419-5653-8>
- Hinkelmann, K., Pierfranceschi, A., & Laurenzi, E. (2016). The Knowledge Work Designer-Modelling Process Logic and Business Logic. In *Lecture Notes in Informatics (LNI), Proceedings - Series of the Gesellschaft für Informatik (GI)* (Vol. P255).

- Hinkelmann, K. (2016). Business Process Flexibility and Decision-Aware Modeling - The Knowledge Work Designer. In *Domain-Specific Conceptual Modeling*, 397–414. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-39417-6_18
- Hinkelmann, K., Gerber, A., Karagiannis, D., Thoenssen, B., van der Merwe, A., & Woitsch, R. (2016). A New Paradigm for the Continuous Alignment of Business and IT: Combining Enterprise Architecture Modelling and Enterprise Ontology. *Computers in Industry*, 79, 77–86. <https://doi.org/10.1016/j.compind.2015.07.009>
- Hinkelmann, K., Laurenzi, E., Lammel, B., Kurjakovic, S., & Woitsch, R. (2016). A Semantically-Enhanced Modelling Environment for Business Process as a Service. In *4th International Conference on Enterprise Systems (ES)*, 143–152. IEEE. <https://doi.org/10.1109/ES.2016.25>
- Hinkelmann, K., Laurenzi, E., Martin, A., & Thönssen, B. (2018). Ontology-Based Metamodeling. In Dornberger R. (Ed.), *Business Information Systems and Technology 4.0. Studies in Systems, Decision and Control*, 177–194. Springer, Cham. https://doi.org/10.1007/978-3-319-74322-6_12
- Hinkelmann, K., Maise, M., & Thönssen, B. (2013). Connecting Enterprise Architecture and Information Objects using an Enterprise Ontology. In *Proceedings of the First International Conference on Enterprise Systems: ES 2013*. IEEE.
- Höfferer, P. (2007). Achieving Business Process Model Interoperability Using Metamodels and Ontologies. In *European Conference on Information Systems*, 1620–1631. University of St. Gallen. <http://aisel.aisnet.org/cgi/viewcontent.cgi?article=1022&context=ecis2007>
- Hölldobler, K., Roth, A., Rumpe, B., & Wortmann, A. (2017). Advances in Modeling Language Engineering, 3–17. Springer, Cham. https://doi.org/10.1007/978-3-319-66854-3_1
- Horkoff, J., Jeusfeld, M. A., Ralyté, J., & Karagiannis, D. (2018). Enterprise Modeling for Business Agility. *Business & Information Systems Engineering*, 60(1), 1–2. <https://doi.org/10.1007/s12599-017-0515-z>
- Horrocks, I., Patel-Schneider, P. F., & van Harmelen, F. (2003). From SHIQ and RDF to OWL: the making of a Web Ontology Language. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(1), 7–26. <https://doi.org/10.1016/J.WEBSEM.2003.07.001>
- Hox, J. J., & Boeije, H. R. (2005). Data collection, primary versus secondary. In K. Kempf-Leonard (Ed.), *Encyclopedia of social measurement* (Vol. 1, pp. 593–599). Elsevier. <https://dspace.library.uu.nl/handle/1874/23634>
- Hrgovic, V., Karagiannis, D., & Woitsch, R. (2013). Conceptual Modeling of the Organisational Aspects for Distributed Applications: The Semantic Lifting Approach. In *2013 IEEE 37th Annual Computer Software and Applications Conference Workshops* (pp. 145–150). IEEE. <https://doi.org/10.1109/COMPSACW.2013.17>
- Hudak, P., & Paul. (1996). Building domain-specific embedded languages. *ACM Computing Surveys*, 28(4), 196. <https://doi.org/10.1145/242224.242477>
- IEEE-SA Standards Board. (2005). *IEEE Standard for Property Specification Language (PSL)*. Institute of Electrical and Electronics Engineers.
- Iivari, J. (2007). A Paradigmatic Analysis of Information Systems as a Design Science. *Scandinavian Journal of Information Systems*, 19(2), 39–64. <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.218.2636>
- Institute of Design Stanford. (2010). *An Introduction to Design Thinking: Process guide*. Stanford University, Palo Alto, USA. <https://dschool-old.stanford.edu/sandbox/groups/designresources/wiki/36873/attachments/74b3d/ModeGuideB OOTCAMP2010L.pdf>
- Izquierdo, J. L. C., Cabot, J., López-Fernández, J. J., Cuadrado, J. S., Guerra, E., & de Lara, J. (2013). Engaging End-Users in the Collaborative Development of Domain-Specific Modelling Languages. In *Cooperative Design, Visualization, and Engineering* (pp. 101–110). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-40840-3_16

- Jablonski, S., Volz, B., & Dornstaeder, S. (2008). A Meta Modeling Framework for Domain Specific Process Management. In *2008 32nd Annual IEEE International Computer Software and Applications Conference* (pp. 1011–1016). IEEE. <https://doi.org/10.1109/COMPSAC.2008.58>
- Jouault, F., Allilaire, F., Bézivin, J., & Kurtev, I. (2008). ATL: A Model Transformation Tool. *Science of Computer Programming*, *72*(1–2), 31–39. <https://doi.org/10.1016/J.SCICO.2007.08.002>
- Jouault F., & Bézivin, J. (2006). KM3: A DSL for Metamodel Specification. In *Proceedings of 8th FMOODS, LNCS 4037*, 171–185. Springer.
- Jun, G. T., Ward, J., Morris, Z., & Clarkson, J. (2009). Health care process modelling: which method when? *International Journal for Quality in Health Care*, *21*(3), 214–224. <https://doi.org/10.1093/intqhc/mzp016>
- Kahn, R., & Cannell, C. (1957). *The Dynamics of Interviewing*. New York: Wiley.
- Kang, K. C., Cohen, S. G., Hess, J. A., Novak, W. E., & Peterson, A. S. (1990). *Feature-Oriented Domain Analysis (FODA) Feasibility Study*. Software Engineering Institute, Carnegie Mellon University. http://www.floppybunny.org/robin/web/virtualclassroom/chap12/s4/articles/foda_1990.pdf
- Kaplan, B., & Maxwell, J. A. (2005). Qualitative Research Methods for Evaluating Computer Information Systems. In J. Anderson & C. Aydin (Eds.), *Evaluating the Organizational Impact of Healthcare Information Systems*, 30–55. Springer New York. https://doi.org/10.1007/0-387-30329-4_2
- Kappel, G., Kapsammer, E., Kargl, H., Kramler, G., Reiter, T., Retschitzegger, W., ... Wimmer, M. (2006). Lifting Metamodels to Ontologies: A Step to the Semantic Integration of Modeling Languages. In *Model Driven Engineering Languages and Systems, Proceedings of the 9th International Conference, MoDELS 2006* (pp. 528–542). Springer, Berlin, Heidelberg. https://doi.org/10.1007/11880240_37
- Karagiannis, D., Mayr, H. C., & Mylopoulos, J. (2016). *Domain-Specific Conceptual Modeling*. (Dimitris Karagiannis, H. C. Mayr, & J. Mylopoulos, Eds.). Cham: Springer International Publishing. <https://doi.org/10.1007/978-3-319-39417-6>
- Karagiannis, D., & Kühn, H. (2002). Metamodelling Platforms. In *Proceedings of the 3rd International Conference EC-Web 2002 -- Dexa 2002, Aix-en-Provence, France, 2002, LNCS 2455* (p. 182). Springer-Verlag.
- Karagiannis, Dimitris. (2015). Agile Modeling Method Engineering. In *Proceedings of the 19th Panhellenic Conference on Informatics - PCI '15*, 5–10. New York, New York, USA: ACM Press. <https://doi.org/10.1145/2801948.2802040>
- Karagiannis, Dimitris. (2018). Conceptual Modelling Methods: The AMME Agile Engineering Approach. In *Informatics in Economy*, 3–19. Springer, Cham. https://doi.org/10.1007/978-3-319-73459-0_1
- Karagiannis, Dimitris, & Buchmann, R. A. (2018). A Proposal for Deploying Hybrid Knowledge Bases: the ADOxx-to-GraphDB Interoperability Case. <https://scholarspace.manoa.hawaii.edu/handle/10125/50399>
- Karagiannis, Dimitris, Buchmann, R. A., Burzynski, P., Reimer, U., & Walch, M. (2016). Fundamental Conceptual Modeling Languages in OMiLAB. In *Domain-Specific Conceptual Modeling*, 3–30. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-39417-6_1
- Karagiannis, Dimitris, & Kühn, H. (2002). Metamodelling Platforms. In K. Bauknecht, A. Min Tjoa, & G. Quirchmayer (Eds.), *Proceedings of the Third International Conference EC-Web at DEXA 2002*. Berlin: Springer-Verlag.
- Karagiannis, Dimitris, & Woitsch, R. (2015). Knowledge Engineering in Business Process Management. In *Handbook on Business Process Management 2*, 623–648. Berlin, Heidelberg:

- Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-45103-4_26
- Keith, B., Vitasek, K., Manrodt, K. B., & Kling, J. (2016). *Strategic Sourcing in the New Economy: Harnessing the Potential of Sourcing Business Models for Modern Procurement* (1st ed.). Palgrave Macmillan US. <https://doi.org/10.1007/978-1-137-55220-4>
- Kelly, S., Lyytinen, K., & Rossi, M. (2013). MetaEdit+ A Fully Configurable Multi-User and Multi-Tool CASE and CAME Environment. In *Seminal Contributions to Information Systems Engineering*, 109–129. Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-36926-1_9
- Kelly, S., & Tolvanen, J.-P. (2008). *Domain-Specific Modeling: Enabling Full Code Generation*. Wiley-Interscience.
- Kidd, P. T. (1994). *Agile Manufacturing: Forging New Frontiers*. Addison-Wesley.
- King, N. (2004). Using Interviews in Qualitative Research. In C. Cassell & G. Symon (Eds.), *Essential Guide to Qualitative Methods in Organizational Research* (p. 408). London: Sage Publications.
- Klein, H. K., & Myers, M. D. (1999). A Set of Principles for Conducting and Evaluating Interpretive Field Studies in Information Systems. *MIS Q*, 23(1), 67–93. <https://doi.org/10.2307/249410>
- Kleppe, A. G. (2009). *Software Language Engineering: Creating Domain-Specific Languages using Metamodels*. Addison-Wesley.
- Kontio, J., Lehtola, L., & Bragge, J. (2004). Using the Focus Group Method in Software Engineering: Obtaining Practitioner and User Experiences. In *Proceedings. 2004 International Symposium on Empirical Software Engineering, 2004. ISESE '04*, 271–280. IEEE. <https://doi.org/10.1109/ISESE.2004.1334914>
- Kopenhagen, N., Gass, O., & Müller, B. (2012). Design Science Research in Action - Anatomy of Success Critical Activities for Rigor and Relevance. In *European Conference on Information Systems (ECIS)*. Association for Information Systems. <https://www.semanticscholar.org/paper/Design-Science-Research-in-Action-Anatomy-of-for-Kopenhagen-Gass/2f8ccdbd88026fb8cac2bd3d24aa5eb5b9f28fe3>
- Kosar, T., Oliveira, N., Mernik, M., João, M., Pereira, V., Črepinšek, M., ... Henriques, P. R. (2010). Comparing General-Purpose and Domain-Specific Languages: An Empirical Study. *Computer Science and Information Systems*, (7(2)), 247–264. <https://doi.org/10.2298/CSIS1002247K>
- Kramler, G., Kappel, G., Reiter, T., Kapsammer, E., Retschitzegger, W., & Schwinger, W. (2006). Towards a Semantic Infrastructure Supporting Model-based Tool Integration. In *GaMMA '06: Proceedings of the 2006 international workshop on Global integrated model management*, 43–46. New York, NY, USA: ACM Press.
- Kritikos, K., Laurenzi, E., & Hinkelmann, K. (2018). Towards Business-to-IT Alignment in the Cloud. In Z. Mann & V. Stolz (Eds.), *Advances in Service-Oriented and Cloud Computing. ESOC 2017. Communications in Computer and Information Science*, 35–52. Springer, Cham. https://doi.org/10.1007/978-3-319-79090-9_3
- Krueger, R. A., & Casey, M. A. (2000). *Focus Group: A Practical Guide for Applied Research* (3rd ed.). Thousand Oaks, CA: Sage Publications.
- Kvale, (1996). *InterViews*. Thousand Oaks, CA: Sage.
- Laurenzi, E., Hinkelmann, K., Reimer, U., Van Der Merwe, A., Sibold, P., & Endl, R. (2017). DSML4PTM: A Domain-Specific Modelling Language for Patient Transferal Management. In *ICEIS 2017 - Proceedings of the 19th International Conference on Enterprise Information Systems* 3, 520–531. Porto, Portugal: SciTePress. <https://doi.org/10.5220/0006388505200531>
- Laurenzi, E. (2014). A Model-Driven Approach to Create and Maintain an Executable Transferal Management Platform. In *Doctoral Consortium - DCSOFT, (ICSOFT 2014)* (pp. 14–20). Vienna, Austria: SciTePress.

<https://www.scitepress.org/PublicationsDetail.aspx?ID=31eQXaM2gLs=&t=1>

- Laurenzi, E., Hinkelmann, K., Goel, M., & Montecchiari, D. (2020). Agile Visualization in Design Thinking. In *New Trends in Business Information Systems and Technology (Accepted for publication)*. Springer Berlin / Heidelberg.
- Laurenzi, E., Hinkelmann, K., Izzo, S., Reimer, U., & van der Merwe, A. (2018). Towards an Agile and Ontology-Aided Modeling Environment for DSML Adaptation. In R. Matulevičius & R. Dijkman (Eds.), *Advanced Information Systems Engineering Workshops. CAiSE 2018. Lecture Notes in Business Information Processing* (pp. 222–234). Springer, Cham. https://doi.org/10.1007/978-3-319-92898-2_19
- Laurenzi, E., Hinkelmann, K., Jüngling, S., Montecchiari, D., Pande, C., & Martin, A. (2019). Towards An Assistive and Pattern Learning-driven Process Modeling Approach. In A. Martin, K. Hinkelmann, A. Gerber, D. Lenat, F. van Harmelen, & P. Clark (Eds.), *Proceedings of the AAAI 2019 Spring Symposium on Combining Machine Learning with Knowledge Engineering (AAAI-MAKE 2019)* (p. 6). Stanford University, Palo Alto, USA: CEUR-WS.org. <http://ceur-ws.org/Vol-2350/paper20.pdf>
- Laurenzi, E., Hinkelmann, K., & van der Merwe, A. (2018). An Agile and Ontology-Aided Modeling Environment. In R. Buchmann, D. Karagiannis, & M. Kirikova (Eds.), *The Practice of Enterprise Modeling. PoEM 2018*. (pp. 221–237). Vienna: Springer, Cham. https://doi.org/10.1007/978-3-030-02302-7_14
- Ledeczki, A., Karsai, G., Maroti, M., Bakay, A., Garrett, J., Thomason, C., ... Volgyesi, P. (2001). The Generic Modeling Environment. In *Proceedings of WISP'2001*. IEEE. <https://www.researchgate.net/publication/233757687>
- Leedy, P. D., & Ormrod, J. E. (2005). *Practical research: Planning and Design*. Prentice Hall.
- Leff, A., & Rayfield, J. T. (2001). Web-Application Development using the Model/View/Controller Design Pattern. In *Proceedings Fifth IEEE International Enterprise Distributed Object Computing Conference*, 118–127. Seattle, WA, USA, USA: IEEE Comput. Soc. <https://doi.org/10.1109/EDOC.2001.950428>
- Lenz, R., Peleg, M., & Reichert, M. (2012). Healthcare Process Support: Achievements, Challenges, Current Research. <http://dbis.eprints.uni-ulm.de/784/>
- Lenz, R., & Reichert, M. (2007). IT support for healthcare processes – premises, challenges, perspectives. *Data & Knowledge Engineering*, 61(1), 39–58. <https://doi.org/10.1016/j.datak.2006.04.007>
- Leppänen, M. (2007). A Context-Based Enterprise Ontology. In *Business Information Systems* (pp. 273–286). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-72035-5_21
- Liao, Y., Lezoche, M., Panetto, H., Boudjlida, N., & Loures, E. R. (2015). Semantic Annotation for Knowledge Explicitation in a Product Lifecycle Management Context: A Survey. *Computers in Industry*, 71, 24–34. Elsevier. <https://doi.org/10.1016/j.compind.2015.03.005>
- Lodderstedt, T., Basin, D., & Doser, J. (2002). SecureUML: A UML-Based Modeling Language for Model-Driven Security, 426–441. Springer.
- Loos, P., Mettler, T., Winter, R., Goeken, M., Frank, U., & Winter, A. (2013). Methodological Pluralism in Business and Information Systems Engineering? The Authors. *Business & Information Systems Engineering*, 5(6), 453–460. <https://doi.org/10.1007/>
- Lynn, T., O'carroll, N., Mooney, J., Helfert, M., Corcoran, D., Hunt, G., ... Healy, P. (2014). Towards a Framework for Defining and Categorizing Business Process-as-a-Service (BPaaS).
- Marca, D., & McGowan, C. L. (1988). *SADT: Structured Analysis and Design Technique*. McGraw-Hill. <https://dl.acm.org/citation.cfm?id=31837>
- March, S. T., & Smith, G. F. (1995). Design and natural science research on information technology.

- Decision Support Systems*, 15(4), 251–266. Elsevier. [https://doi.org/10.1016/0167-9236\(94\)00041-2](https://doi.org/10.1016/0167-9236(94)00041-2)
- March, & Storey. (2008). Design Science in the Information Systems Discipline: An Introduction to the Special Issue on Design Science Research. *MIS Quarterly*, 32(4), 725. <https://doi.org/10.2307/25148869>
- Marshall, C., & Rossman, G. B. (1999). *Designing Qualitative Research*. Thousand Oaks, CA: Sage.
- Martin, Andreas. (2016). *A Combined Case-based Reasoning and Process Execution Approach for Knowledge-Intensive Work*. University of South Africa. <http://hdl.handle.net/10500/22796>
- Martin, J., & James. (1983). *Managing the Data-base Environment*. Prentice-Hall. <https://dl.acm.org/citation.cfm?id=538746>
- Mathe, J. L., Martin, J. B., Miller, P., Lédeczi, Á., Weavind, L. M., Nadas, A., ... Sztipanovits, J. (2009). A Model-Integrated, Guideline-Driven, Clinical Decision-Support System. *IEEE Software*, 26(4), 54–61. <https://doi.org/10.1109/MS.2009.84>
- Maxwell, J. A. (2013). *Qualitative Research Design: An Interactive Approach* (3rd ed.). Sage Publications. <http://www.sagepub.com/textbooks/Book234502>
- McGuinness, D. L., & van Harmelen, F. (2014). *OWL Web Ontology Language Overview*. <https://www.w3.org/TR/2004/REC-owl-features-20040210/#s6>
- Mernik, M., Hering, J., & Sloane, A. M. (2005). When and How to Develop Domain-Specific Languages. *ACM Computing Surveys*, 37(4), 316–344. <https://doi.org/10.1145/1118890.1118892>
- Merriam, S. B. (2009). *Qualitative Research: A Guide to Design and Implementation*. San Francisco: Jossey-Bass. <http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0470283548.html>
- Merriam, Sharan B. (1998). *Qualitative Research and Case Study Applications in Education* (2nd ed.). San Francisco, CA: Jossey-Bass Publishers.
- Miles, M. B., & Huberman, A. M. (1994). *Qualitative Data Analysis* (2nd ed.). Thousand Oaks, CA: Sage.
- Miller Joaquin Mukerji, J. (2003). MDA Guide Version 1.0.1. (J. Miller & J. Mukerji, Eds.). Needham and Massachusetts. <http://www.omg.org/cgi-bin/doc?omg/03-06-01>
- Miron, E.-T., Muck, C., & Karagiannis, D. (2019). Transforming Haptic Storyboards into Diagrammatic Models: The Scene2Model Tool. In *Proceedings of the 52nd Hawaii International Conference on System Sciences*. Hawaii. <https://hdl.handle.net/10125/59494>
- Moore, B. (2004). *Eclipse development using the graphical editing framework and the eclipse modeling framework*. IBM, International Technical Support Organization.
- Mouton, J. (2001). *How to Succeed in your Master's and Doctoral Studies: A South African Guide and Resource Book*. Pretoria: Van Schaik.
- Myers, M. D. (2009). *Qualitative Research in Business and Management*. London: Sage.
- National Center for Biomedical Ontology. (2012). International Classification of Functioning, Disability and Health - Summary | NCBO BioPortal. Retrieved 23 January 2019, from <https://bioportal.bioontology.org/ontologies/ICF>
- Natschläger, C. (2011). Towards a BPMN 2.0 ontology. In *Lecture Notes in Business Information Processing*, 1–15. Springer Verlag. https://doi.org/10.1007/978-3-642-25160-3_1
- Neumann, J., Rockstroh, M., Franke, S., & Neumuth, T. (2016). BPMNSIX – A BPMN 2.0 Surgical Intervention Extension: Concept and Design of a BPMN Extension for Intraoperative Workflow Modeling and Execution in the Integrated Operating Room. In *7th Workshop on Modeling and Monitoring of Computer Assisted Interventions (M2CAI) - 19th International Conference on Medical Image Computing and Computer Assisted Interventions (MICCAI 2016), At Athens, Greece*.

- Newton, S. W. (2012). *Introduction to Educational Research: A Critical Thinking Approach* (2nd ed.). Arkansas: Sage Publications.
<http://www.sagepub.com/books/Book235696/toc#tabview=toc>
- Nicola, A. De, Mascio, T. Di, Lezoche, M., & Tagliano, F. (2008). Semantic Lifting of Business Process Models. In *2008 12th Enterprise Distributed Object Computing Conference Workshops* (pp. 120–126). IEEE. <https://doi.org/10.1109/EDOCW.2008.55>
- Nikles, S., & Brander, S. (2009). Separating Conceptual and Visual Aspects in Meta-Modelling. In *Proceedings of the Joint Workshop on Advanced Technologies and Techniques for Enterprise Information Systems*. 1, 90–94. Scitepress.
<http://www.scitepress.org/DigitalLibrary/PublicationsDetail.aspx?ID=26AZReQIP4A=&t=1>
- Nonaka, I., Toyama, R., & Nagata, A. (2000). A Firm as a Knowledge-creating Entity: A New Perspective on the Theory of the Firm. In I. Nonaka, R. Toyama, & A. Nagata (Eds.), *Industrial and corporate change*, 1–20. Oxford University Press.
https://ai.wu.ac.at/~kaiser/seiw/Nonaka_ICC_2000.pdf
- Nunes, D. A., & Schwabe, D. (2006). Rapid Prototyping of Web Applications Combining Domain Specific Languages and Model Driven Design. In *Proceedings of the 6th International Conference on Web Engineering* (pp. 153–160). New York and NY and USA: ACM.
<https://doi.org/10.1145/1145581.1145616>
- OMG. (2006). *Organization Structure Metamodel (OSM)*. <https://www.omg.org/cgi-bin/doc?bmi/09-08-02>
- OMG. (2011). Business Process Model and Notation (BPMN), Version 2.0. Object Management Group OMG.
<http://www.omg.org/spec/BPMN/20100501>
<http://www.omg.org/spec/BPMN/20100502>
- OMG. (2014a). Business Motivation Model. *BMM 1.2*. <https://doi.org/formal/2008-08-02>
- OMG. (2014b). Object Constraint Language. <http://www.omg.org/spec/OCL/2.4>
- OMG. (2015). *Semantics of Business Vocabulary and Business Rules (SBVR)*.
<http://www.omg.org/spec/SBVR/1.3/PDF>
- OMG. (2016a). Case Management Model and Notation (CMMN V 1.1).
<http://www.omg.org/spec/CMMN/1.1/PDF/>
- OMG. (2016b). *Decision Model and Notation*. <http://www.omg.org/spec/DMN/1.1/PDF/>
- OMG. (2016c). *Meta Object Facility (MOF) Core Specification* (OMG Available Specification). Object Management Group. <http://www.omg.org/spec/MOF/2.5.1>
- OMG. (2017). *Unified Modeling Language 2.5.1*. <https://www.omg.org/spec/UML/2.5.1>
- OMG. (2018). *Value Delivery Modeling Language Specification Version 1.1*.
<https://www.omg.org/spec/VDML>
- Open Group. (2017). ArchiMate® 3.0.1 Specification.
<http://pubs.opengroup.org/architecture/archimate3-doc/chap15.html>
- Parreiras, F. S. (2012). *Semantic Web and Model-Driven Engineering*. Hoboken, NJ, USA: John Wiley & Sons, Inc.
- Parry, C., Mahoney, E., Chalmers, S. A., & Coleman, E. A. (2008). Assessing the Quality of Transitional Care. *Medical Care*, 46(3), 317–322.
<https://doi.org/10.1097/MLR.0b013e3181589bdc>
- Patton, M. Q. (2015). *Qualitative research & Evaluation Methods: Integrating Theory and Practice* (4th ed.). SAGE Publications.
- Peffer, K., Rothenberger, M., Tuunanen, T., & Vaezi, R. (2012). Design Science Research Evaluation. In *Proceedings of the 7th international conference on Design Science Research in Information Systems: advances in theory and practice*, 398–410. Springer-Verlag.

https://doi.org/10.1007/978-3-642-29863-9_29

- Pérez, B., & Porres, I. (2013). Reasoning about UML/OCL Models using Constraint Logic Programming and MDA. In *ICSEA 2013, The Eighth International Conference on Software Engineering Advances*, 228–233. http://www.thinkmind.org/index.php?view=article&articleid=icsea_2013_8_10_10352
- Pérez, F., Valderas, P., & Fons, J. (2011). Towards the Involvement of End-Users within Model-Driven Development. In *End-user Development* (pp. 258–263). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-21530-8_23
- Petre, M. (2013). UML in Practice. In *ICSE '13 Proceedings of the 2013 International Conference on Software Engineering* (pp. 722–731). San Francisco, CA, USA : IEEE Press Piscataway, NJ, USA. <https://doi.org/978-1-4673-3076-3>
- Poppendieck, M. (2007). Lean Software Development. In *ICSE COMPANION '07: Companion to the proceedings of the 29th International Conference on Software Engineering*, 165–166. <https://doi.org/10.1109/ICSECOMPANION.2007.46>
- Pries-Heje, J., Baskerville, R., & Venable, J. (2008). Strategies for Design Science Research Evaluation. In *Conference Proceedings, 16th European Conference on Information Systems*. National University of Ireland. <https://forskning.ruc.dk/en/publications/strategies-for-design-science-research-evaluation>
- Ranabahu, A. H., Sheth, A. P., Manjunatha, A., Thirunarayan, K., Ranabahu, A., & Sheth, A. (2012). Towards Cloud Mobile Hybrid Application Generation using Semantically Enriched Domain Specific Languages. *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, 76, 349–360. <http://corescholar.libraries.wright.edu/knoesis>
- Reimer, U., & Laurenzi, E. (2014). Creating and maintaining a collaboration platform via domain-specific reference modelling. In *EChallenges e-2014 Conference: 29-30 October 2014, Belfast, Ireland*, 1–9. IEEE.
- Robert, S., Gérard, S., Terrier, F., & Lagarde, F. (2009). A Lightweight Approach for Domain-Specific Modeling Languages Design. <https://doi.org/10.1109/SEAA.2009.81>
- Robol, M., Salnitri, M., & Giorgini, P. (2017). Toward GDPR-compliant socio-technical systems: Modeling language and reasoning framework. In *Lecture Notes in Business Information Processing*, 305, 236–250. Springer Verlag. https://doi.org/10.1007/978-3-319-70241-4_16
- Robson, C. (2002). *Real World Research* (2nd edn). Oxford: Blackwell.
- Rodrigues, A., & Silva, D. (2015). Model-Driven Engineering: A Survey Supported by the Unified Conceptual Model. <https://doi.org/10.1016/j.cl.2015.06.001>
- Rose, L. M., Kolovos, D. S., Paige, R. F., & Polack, F. A. C. (2009). Enhanced Automation for Managing Model and Metamodel Inconsistency. In *2009 IEEE/ACM International Conference on Automated Software Engineering* (pp. 545–549). IEEE. <https://doi.org/10.1109/ASE.2009.57>
- Rospoche, M., Ghidini, C., & Serafini, L. (2014). An Ontology for the Business Process Modelling Notation. In *8th International Conference on Formal Ontology in Information Systems*. <https://dkm-static.fbk.eu/people/rosbacher/files/pubs/2014foisbpmn.pdf>
- Salehi, P., Hamou-Lhadj, A., Toeroe, M., & Khendek, F. (2016). A UML-based Domain Specific Modeling Language for Service Availability Management: Design and experience. *Computer Standards & Interfaces*, 44, 63–83. <https://doi.org/10.1016/j.csi.2015.09.009>
- Sandkuhl, K., Fill, H.-G., Hoppenbrouwers, S., Krogstie, J., Matthes, F., Opdahl, A., ... Winter, R. (2018). From Expert Discipline to Common Practice: A Vision and Research Agenda for Extending the Reach of Enterprise Modeling. *Business & Information Systems Engineering*, 60(1), 69–80. <https://doi.org/10.1007/s12599-017-0516-y>
- Saunders, M., Lewis, P., & Thornhill, A. (2009). *Research Methods for Business Students*. Harlow

- and England: Pearson Education.
- Saunders, M. N. K., Lewis, P., & Thornhill, A. (2019). *Research Methods for Business Students* (8th ed.). Harlow, UK: Pearson Education Limited.
- Scheer, A.-W., & Nüttgens, M. (2000). ARIS Architecture and Reference Models for Business Process Management, 376–389. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-45594-9_24
- Schmidt, D. C. (2006). Model-Driven Engineering. *IEEE Computer*, 39(2), 25–31. <http://www.computer.org/portal/site/computer/menuitem.e533b16739f5>
- Schwaber, K., & Beedle, M. (2002). *Agile Software Development with Scrum*. Prentice Hall - Computers. http://sutlib2.sut.ac.th/sut_contents/H129174.pdf
- Selic, B. (2007). A Systematic Approach to Domain-Specific Language Design Using UML. In *10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC'07)*. IEEE. <https://pdfs.semanticscholar.org/2460/618352d17adf34ce99544f2fc0ad59c58019.pdf>
- Selic, B. (2011). The Theory and Practice of Modeling Language Design for Model-Based Software Engineering—A Personal Perspective, 290–321. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-18023-1_7
- Shenvi, R., Shenvi, R., Mazza, G., Saini, D., Orthner, H., & Gray, J. (2007). Generation of Context-Specific Electronic Patient Care Reports (ePCR) using Domain-Specific Modeling. <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.131.5832>
- Shneiderman, B., & Plaisant, C. (2004). *Designing the User Interface: Strategies for Effective Human-Computer Interaction* (4th Edition). <http://dl.acm.org/citation.cfm?id=983803>
- Silver, B. (2011). *BPMN Method and Style, Second Edition* (Second Ed.). Aptos, CA: Cody-Cassidy Press.
- Silver, B. (2011). *BPMN Method and Style: with BPMN Implementer's Guide*. Cody-Cassidy Press.
- Silverman, D. (2007). *A Very Short, Fairly Interesting and Reasonably Cheap Book about Qualitative Research*. London: Sage Publications.
- Simon, H. A. (1996). *The Sciences of the Artificial* (3rd ed.). Cambridge and MA and USA: MIT Press.
- Sonnenberg, C., & vom Brocke, J. (2012). Evaluation Patterns for Design Science Research Artefacts. In M. Helfert & B. Donnellan (Eds.), *Practical Aspects of Design Science* 71–83. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-33681-2_7
- Staab, S., & Studer, R. (2009). *Handbook on Ontologies*. Springer.
- Stahl, T., & Völter, M. (2006). *Model-Driven Software Development: Technology, Engineering, Management*. Chichester and England and Hoboken and NJ: John Wiley.
- Stake, R. E. (1995). *The Art of Case Study Research*. Thousand Oaks, CA: SAGE Publications.
- Stewart, D. W., Shamdasani, P. N., & Rook, D. W. (2007). *Focus Groups: Theory and Practice* (2nd ed.). *Applied Social Research Methods Series*, 20. Newbury Park, CA: Sage Publications.
- Strauss, A., & Corbin, J. M. (2015). *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory* (4th ed.). Thousand Oaks, CA: Sage.
- Strembeck, M., & Zdun, U. (2009). An Approach for the Systematic Development of Domain-Specific Languages. *Software - Practice and Experience*, 39, 1253–1292. <https://doi.org/10.1002/spe.936>
- Stroppi, L. J. R., Chiotti, O., & Villarreal, P. D. (2011). Extending BPMN 2.0: Method and Tool Support. In *Business Process Model and Notation* (pp. 59–73). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-25160-3_5

- Studer, R., Benjamins, V. R., & Fensel, D. (1998). Knowledge Engineering: Principles and Methods. *Data & Knowledge Engineering*, 25(1–2), 161–197. [https://doi.org/10.1016/S0169-023X\(97\)00056-6](https://doi.org/10.1016/S0169-023X(97)00056-6)
- Sutii, A. M., Verhoeff, T. & Van Den Brand, M. G. J. (2014). Ontologies in Domain Specific Languages: A Systematic Literature Review. Computer science reports, 1409. Eindhoven: Technische Universiteit. <https://pure.tue.nl/ws/files/3889700/353021132621361.pdf>
- Syriani, E., Vangheluwe, H., Mannadiar, R., Hansen, C., Van Mierlo, S., & Ergin, H. (2013). AToMPM: A Web-based Modeling Environment. In *CEUR Workshop Proceedings*, 1115, 21–25.
- Tashakkori, A., & Teddie, C. (2003). *Handbook of Mixed Methods in Social and Behavioral Research*. Thousand Oaks, CA: Sage.
- The Open Group. (2011). TOGAF® Version 9.1. *Van Haren Publishing*.
- The Open Group. (2012a). ArchiMate. <http://pubs.opengroup.org/architecture/archimate2-doc/>
- The Open Group. (2012b). ArchiMate 2.1 Specification.
- The Open Group. (2017). ArchiMate® 3.0.1 Specification. <http://pubs.opengroup.org/architecture/archimate3-doc/>
- Thomas, O., & Fellmann, M. (2007). Semantic EPC: Enhancing Process Modeling Using Ontology Languages. *SBPM*. <https://www.semanticscholar.org/paper/Semantic-EPC%3A-Enhancing-Process-Modeling-Using-Thomas-Fellmann/c065ed04ecf0db1d5e548cb99731fd1154976236>
- Thönssen, B., & Lutz, J. (2013). Semantically Enriched Obligation Management: An Approach for Improving the Handling of Obligations Represented in Contracts. In A. Fred, J. L. G. Dietz, K. Liu, & J. Filipe (Eds.), *Knowledge Discovery, Knowledge Engineering and Knowledge Management. IC3K 2012. Communications in Computer and Information Science*, 415, 337–349. Springer Verlag. https://doi.org/10.1007/978-3-642-54105-6_23
- Tullis, T. (Thomas), & Albert, B. (William). (2013). *Measuring the User Experience: Collecting, Analyzing, and Presenting Usability Metrics*. Elsevier.
- Uschold, M., & Grüninger, M. (1996). *Ontologies: Principles, Methods and Applications. Technical Report - University of Edinburgh Artificial Intelligence Applications Institute AIAI TR*.
- Uschold, M., King, M., Morale, S., & Zorgios, Y. (1998). The Enterprise Ontology. *The Knowledge Engineering Review*, 13(01), 31–89.
- Vaishnavi, V., & Kuechler, B. (2004). Design Science Research in Information Systems. *Journal MIS Quarterly*, 28(1), 75–105. <http://desrist.org/desrist/content/design-science-research-in-information-systems.pdf>
- van Deursen, A., Klint, P., & Visser, J. (2000). Domain-Specific Languages: An Annotated Bibliography. *SIGPLAN Not*, 35(6), 26–36. <https://doi.org/10.1145/352029.352035>
- Van Harmelen, F., Lifschitz, V., & Porter, B. (2008). *Handbook of knowledge representation*. Elsevier. <https://www.sciencedirect.com/bookseries/foundations-of-artificial-intelligence/vol/3>
- Van Harmelen, F., & Ten Teije, A. (2019). A Boxology of Design Patterns for Hybrid Learning and Reasoning Systems. *Journal of Web Engineering*, 18(1–3), 97–124. <https://doi.org/10.13052/jwe1540-9589.18133>
- Vernadat, F. B. (2003). Enterprise Modelling and Integration. In Kosanke K., Jochem R., Nell J.G., Bas A.O. (eds) *Enterprise Inter- and Intra-Organizational Integration. ICEIMT 2002. IFIP — The International Federation for Information Processing*, 108. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-35621-1_4
- Visual Paradigm. (2018). Chapter 4. Profile and Stereotype - Visual Paradigm Community Circle. Retrieved 26 October 2019, from <https://circle.visual-paradigm.com/docs/profile-and-stereotype/>
- Völter, M., Stahl, T., Bettin, J., Haase, A., & Helsen, S. (2013). *Model-Driven Software Development:*

- Technology, Engineering, Management*. West-Sussex, England: John Wiley & Sons.
- von Eiff, W., Schüring, S., Greitemann, B., & Karoff, M. (2011). REDIA – Auswirkungen der DRG-Einführung auf die Rehabilitation. *Die Rehabilitation*, 50(04), 214–221. <https://doi.org/10.1055/s-0031-1275720>
- von Halle, B., & Goldberg, L. (2010). *The Decision Model: A Business Logic Framework Linking Business and Technology*. CRC Press Auerbach Publications.
- W3C (2004a). RDF Semantics. <https://www.w3.org/TR/2004/REC-rdf-nt-20040210/>
- W3C (2004b). Resource Description Framework (RDF): Concepts and Abstract Syntax. <https://www.w3.org/TR/rdf-concepts/>
- W3C (2008a). SPARQL Query Language for RDF. <https://www.w3.org/TR/rdf-sparql-query/>
- W3C (2008b). *SPARQL Update - A Language for Updating RDF Graphs*. <https://www.w3.org/Submission/SPARQL-Update/>
- W3C (2011). SPIN - SPARQL Syntax. <https://www.w3.org/Submission/spin-sparql/>
- W3C (2012). W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes. <https://www.w3.org/TR/xmlschema11-2/>
- W3C (2014a). RDF 1.1 Concepts and Abstract Syntax. <https://www.w3.org/TR/rdf11-concepts/>
- W3C (2014b). RDF 1.1 Semantics. <https://www.w3.org/TR/rdf11-nt/>
- W3C (2014c). RDF Schema 1.1. <https://www.w3.org/TR/rdf-schema/>
- W3C (2017). Shapes Constraint Language (SHACL). <https://www.w3.org/TR/shacl/>
- W3C OWL Working Group. (2009). OWL Web Ontology Language Overview.
- Walker, D., & Betz, P. (2013). *The Emergency Flow Concept* (1st Ed.). Zurich, Switzerland: Walkerproject AG.
- Walter, T., Parreiras, F. S., & Staab, S. (2014). An Ontology-based Framework for Domain-Specific Modeling. *Software & Systems Modeling*, 13(1), 83–108. <https://doi.org/10.1007/s10270-012-0249-9>
- Wegeler, T., Gutzeit, F., Destailleur, A., & Dock, B. (2013). Evaluating the Benefits of Using Domain-Specific Modeling Languages. In *Proceedings of the 2013 ACM workshop on Domain-specific modeling - DSM '13*, 7–12. New York, New York, USA: ACM Press. <https://doi.org/10.1145/2541928.2541930>
- Woitsch, R., Hinkelmann, K., Maria, A., Ferrer, J., & Yuste, J. I. (2016). Business Process as a Service (BPaaS): The BPaaS Design Environment. <http://ceur-ws.org/Vol-1600/session1p1.pdf>
- Woitsch, R., & Utz, W. (2015). Business Processes as a Service (BPaaS): A Model-Based Approach to Align Business with Cloud Offerings. *eChallenges e-2015 Conference*, Vilnius, 2015, 1-8. IEEE.
- Wolcott, H. (2001). *Writing Up Qualitative Research*. Thousand Oaks, CA: Sage Publications.
- World Health Organization. (2016). *WHO | International Classification of Functioning, Disability and Health (ICF)*. WHO. World Health Organization. <http://www.who.int/classifications/icf/en/>
- Wu, Y., Allen, A. A., Hernandez, F., France, R., & Clarke, P. J. (2012). A Domain-Specific Modeling Approach to Realizing User-Centric Communication. *Software: Practice and Experience*, 42(3), 357–390. ACM. <https://doi.org/10.1002/spe.1081>
- Wüest, D., Seyff, N., & Glinz, M. (2017). *FlexiSketch: a Lightweight Sketching and Metamodeling Approach for End-Users*. *Software & Systems Modeling*. Springer Berlin Heidelberg. <https://doi.org/10.1007/s10270-017-0623-8>
- Yazan, B. (2015). Three Approaches to Case Study Methods in Education: Yin, Merriam, and Stake. *The Qualitative Report*, 20(2), 134–152. <https://nsuworks.nova.edu/tqr/vol20/iss2/12>

- Yin, R. K. (2003). *Case Study Research: Design and Methods*. Thousand Oaks, CA: Sage Publications.
- Zachman, J. A. (2008). The Concise Definition of The Zachman Framework by: John A. Zachman. <https://www.zachman.com/about-the-zachman-framework>
- Zečević, I., Bjeljic, P., Perišić, B., Maruna, V., & Venus, D. (2017). Domain-Specific Modeling Environment for Developing Domain Specific Modeling Languages as Lightweight General-Purpose Modeling Language Extensions, 872–881. Springer, Cham. https://doi.org/10.1007/978-3-319-56535-4_85
- Zhou, J., Zhao, D., & Liu, J. (2011). A Domain Specific Language for Interactive Enterprise Application Development. In Z. Gong, X. Luo, J. Chen, J. Lei, & F. Wang (Eds.), *Web Information Systems and Mining*, 6988, 351–360. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-23982-3_43

ABBREVIATIONS AND ACRONYMS

Table 67 Abbreviation overview

Abbreviation / Acronym	Extended name
AI	Artificial Intelligence
BPaaS	Business Process as a Service
BPM	Business Process Management
BPMN	Business Process Model and Notation
CMMN	Case Management Model and Notation
CP	Clinical Pathway
DMN	Decision Model and Notation
DSL	Domain-Specific Language
DSML	Domain-Specific Modelling Language
DSML4PTM	Domain-Specific Modelling Language for Patient Transferal Management
DSR	Design Science Research
EDI	Electronic Data Interchange
EPC	Event-driven Process Chain
ePCR	Electronic Patient Care Record
ESI	Emergency Severity Index

FHNW	Fachhochschule Nordwestschweiz (University of Applied Science and Arts Northwestern Switzerland)
FHSG	Fachhochschule St. Gallen (University of Applied Sciences St. Gallen)
GPML	Generic Purpose Modelling Language
ICD	International Statistical Classification of Diseases and Related Health Problems
ICF	International Classification of Functioning, Disability and Health
KoGu	KostenGutsprache (cost reimbursement)
MDE	Model-Driven Engineering
OMG	Object Management Group
UML	Unified Modelling Language
XML	eXtensible Markup Language

LIST OF FIGURES

Figure 1. The AMME Lifecycle (extracted from Bork et al., 2019)	6
Figure 2. Map of the Consolidated Research Problem	9
Figure 3. Thesis chapter layout	14
Figure 4. Components of modelling methods. Three modelling language specifications within the red box (Karagiannis & Kühn, 2002).....	18
Figure 5. Definition of levels by MOF (OMG 2016c)	19
Figure 6. Meta-modelling hierarchy (Strahringer, 1996).....	20
Figure 7. Representation of standard modelling languages, adapted from (Efendioglu et al., 2017)	21
Figure 8. Abstraction layers of BPMN in meta-modelling (Karagiannis <i>et al.</i> , 2016).....	22
Figure 9. Language Barrier between Application Domain and Information Technology (Frank, 2010)	23
Figure 10. DSMLs vs. GPMLs adapted from (Frank, 2013b).....	26
Figure 11. The four dimensions of a knowledge space (Karagiannis & Woitsch, 2015).....	29
Figure 12. Integrated modelling languages in the Knowledge Work Designer modelling tool. Adapted from (Hinkelmann, 2016).....	30
Figure 13. Typical modelling approach: separating models for human and machine interpretation (Hinkelmann et al., 2018).....	31
Figure 14. Comparison between "domain-specific adaptation" of BPMN and "from scratch" approach to develop DSMLs (Braun et al., 2015)	34
Figure 15. The abstract syntax of DR DSML (Chiprianov et al., 2013).....	35
Figure 16. The concrete syntax of the DR DSML (Chiprianov et al., 2013).....	35
Figure 17. Extensibility of abstract and concrete syntax within MOF-based languages and their provision of methodical support for extension design (Braun, 2015b).....	37
Figure 18. DSMLs: Potential Productivity Gain vs. Scale of Reuse (Frank, 2010)	41
Figure 19. DSL engineering lifecycle (Ceh et al., 2011)	44
Figure 20. Meta-model Engineering Lifecycle (Izquierdo et al., 2013)	45
Figure 21. Typical DSML engineering life-cycle (Barišić et al., 2018)	46
Figure 22. DSML Development Process (Cho et al., 2012)	47
Figure 23 The seven steps in the macro process (Frank, 2013a)	48
Figure 24. The AMME (or OMiLab) Lifecycle. Taken from (Bork et al., 2019; Karagiannis, 2018; Efendioglu et al., 2017; Karagiannis 2015)	50
Figure 25. Architecture for semantic interoperability using both metamodels and ontologies	59
Figure 26. The “research onion”. Adapted from (Saunders, Lewis & Thornihill, 2009)	62
Figure 27 The adopted research approach. Adapted from (Saunders et al., 2009).....	67
Figure 28. Design science research conceptual framework and cycles. Adapted from (Hevner, 2007)	69

Figure 29. Reasoning in the general design cycle (GDC) (Vaishnavi and Kuechler, 2004) ...	69
Figure 30. Design Science Research (DSR) methodology applied in this research work. Adapted from Vaishnavi and Kuechler (2004) and DSR cycles from Hevner (2007)	74
Figure 31. Methodological choices (adapted from Saunders et al. 2019)	78
Figure 32. Source triangulation and method triangulation employed in this research	81
Figure 33. Instantiation of the AMME Lifecycle (Karagiannis, 2015; Karagiannis, 2018) for creating DSML4PTM	91
Figure 34. Compliance of requirements of the modelling approaches (Burwitz et al., 2013).	94
Figure 35 CP-Mod meta model excerpt – Evidence-based Decision and categorised Action (Burwitz et al., 2013)	95
Figure 36. CP model using CP-mod (Burwitz et al., 2013).....	95
Figure 37 Compliance of requirements of the modelling approaches (Heß et al., 2015).....	96
Figure 38. Meta-model excerpt of DSML4CP (Heß et al., 2015)	97
Figure 39. Excerpt from the Soft Tissue Sarcoma clinical pathway process model (Heß et al., 2015)	98
Figure 40. Procedure for BPMN extensions (Braun et al., 2015).....	99
Figure 41. An excerpt of the BPMN4CP meta-model.....	100
Figure 42. A wisdom tooth treatment process model built with BPMN4CP (Braun et al., 2015)	100
Figure 43. Integrated BPMN extension method (Braun et al., 2016).....	101
Figure 44. Abstract syntax of BPMN4CP 2.0 (Braun et al., 2016)	101
Figure 45. Stroke scenario modelled with BPMN4CP 2.0 (Braun et al., 2016).....	102
Figure 46. An excerpt of the meta-model of BPMNsix (Braun et al., 2015).....	103
Figure 47. An excerpt of the concrete syntax for BPMNsix.....	104
Figure 48. Instantiation of the Create phase for DSML4PTM: In depth domain analysis	107
Figure 49. An excerpt of the conceptual model derived from the Information Model Document	109
Figure 50. Versions of the Reference Process Models: from the first (v1) to the last version (v11).....	110
Figure 51. Changes in the Reference Process Model: version 5 vs. version 11	111
Figure 52. Intertwin between the domain analysis and the requirement elicitation activities	116
Figure 53. Mock-up for the process activity “Prepare KoGu”	116
Figure 54. Reference Process Model Activities.....	117
Figure 55. Instantiation of the Design phase: the DSML4PTM meta-model	122
Figure 56. Steps to design the meta-model of DSML4PTM	123
Figure 57. Example of prescribed flow in the reference process.....	124
Figure 58. Extended BPMN.....	125
Figure 59. Relevant events according to requirements. Adapted from (Silver, 2011).	126
Figure 60. Extended CMMN, extended Control Element Meta-Model and their relation	128

Figure 61. Example of decisions logic modelled in the reference process	129
Figure 62. Extended DMN and references with other meta-models	130
Figure 63. Example of status and version on extended data objects	132
Figure 64. Extended Document and Knowledge Meta-Model and references to other meta-models	133
Figure 65. Extended Organisational Meta-Model and references to BPMN.....	134
Figure 66. Development phase instantiation for DSML4PTM.....	138
Figure 67. Result of implementation of task types and names	140
Figure 68. Propagation of changes from the Develop phase of the AMME Lifecycle	141
Figure 69. Deploy/Validate phase instantiation for DSML4PTM.....	142
Figure 70. Propagation of changes from the Deploy/Validate phase of the AMME Lifecycle	143
Figure 71. Part of the reference model implemented in the ADOxx Development Toolkit..	147
Figure 72. Components of the BPaaS Design Environment.....	152
Figure 73. Two AMME Lifecycle instantiations for the human and machine interpretation of BPaaS DSML.....	154
Figure 74. Send Invoice Business Process.....	160
Figure 75. A workflow implementing the business process Send Invoice	162
Figure 76. The BPaaS Meta-Models.....	163
Figure 77. BPaaS Business Process Modelling Language Class Diagram.....	164
Figure 78. Semantic annotation of models	165
Figure 79. Functional Description dimension implemented in ADOxx	166
Figure 80. First Iteration: Implementation of DSML BPaaS in ADOxx.....	167
Figure 81. Business Process Requirement modelling view vs. Workflow Description modelling view.....	168
Figure 82. Adaptation of the “Group” concept to specify business process requirements....	170
Figure 83. Adaptation of the Lane to specify workflow descriptions.....	171
Figure 84. Example of Business Process Functional Requirements annotated with APQC, Object and Action	173
Figure 85. Selection box with the APQC categories of the first tier	174
Figure 86. Excerpt of the APQC Ontology.....	175
Figure 87. Excerpt of the FBPDO Ontology	176
Figure 88. Performance dimension in business terms	178
Figure 89. Performance dimension in IT terms	179
Figure 90. Search-insert type function applied on the APQC question.....	181
Figure 91. Hierarchical cloud service identification employing the questionnaire	182
Figure 92. Excerpt of the Questionnaire Ontology	183
Figure 93. Business process non-functional requirement specification through the questionnaire (Kritikos et al., 2018).....	183
Figure 94. Main problems hindering agility of DSML engineering faced during the development of DSML4PTM and BPaaS DSML.....	186

Figure 95. The separation between the language engineering component and the modelling component.....	186
Figure 96. Adaptation Complexity Levels.....	195
Figure 97. Meta-modelling vs. an agile meta-modelling.....	200
Figure 98. Integrating language engineering and modelling into the same component	202
Figure 99. Sketch of an Agile Meta-Modelling.....	203
Figure 100. Semantic Specification Components from Modelling Method Components (Karagiannis & Kühn 2002).....	204
Figure 101. Derivation of Operators.....	206
Figure 102 The 10 operators for on-the-fly Domain-Specific Adaptations.....	210
Figure 103. Alignment between meta-model and ontology; and transformation of models into ontology instances.....	213
Figure 104. Inconsistency between the human- and machine-interpretable representations of modelling languages in the agile meta-modelling approach. Adapted from (Höfferer, 2007).	214
Figure 105. Ontology-based meta-modelling (Hinkelmann, Laurenzi et al. 2018).....	217
Figure 106. An ontology-based meta-modelling for a context-adaptive questionnaire.....	218
Figure 107. Architecture of the Ontology-Aided Approach.....	221
Figure 108. Ontology-based Meta-Model.....	222
Figure 109. BPMN graphical notations grouped by categories in the Bizagi Modeller.....	224
Figure 110. Implemented ontologies and their interlinkage	226
Figure 111. Semantic rules for the propagation of changes from the human to the machine-interpretable representation.....	230
Figure 112. Methodology to design and evaluate semantic rules for the propagation domain-specific adaptations. Adapted from (Grüniger & Fox, 1995)	231
Figure 113. Syntactic Validation of SPARQL Rule 1	235
Figure 114 Conceptualisation of the use case for the integration of CMMN Discretionary Task with BPMN Manual Task	236
Figure 115. Semantic validation of SPARQL Rule 1 - modelling elements integration.....	237
Figure 116. “As-is” use case for the extension of the Data_Document concept in the Document and Knowledge Meta-Model	244
Figure 117. Results of instantiation of SPARQL rules 2 to 5.....	250
Figure 118. Components of the AOAME architecture	264
Figure 119. Model-View-Controller (MVC) design pattern in AOAME.....	268
Figure 120. Three steps to populate the Palette	270
Figure 121. Correspondence between the four endpoints and the ontology structure.....	271
Figure 122. Snippets of the Java methods that generate SPARQL SELECTS for retrieving (1) modelling languages and (2) modelling view(s).....	272
Figure 123. Snippets of the Java methods that generate SPARQL SELECTS for retrieving (3) categories and (4) graphical notations	273
Figure 124. Execution of a concrete query generated from the fourth endpoint “getPaletteElement”	274

Figure 125. Main functionalities in the palette of AOAME	275
Figure 126. Main view for extending a modelling construct: Creation of the new class and its annotation properties.....	277
Figure 127. Excerpt of the method implemented in the Web service to generate the SPARQL INSERT DATA.....	279
Figure 128. Adding datatype properties and object properties (i.e. bridging connectors and semantic mappings).....	280
Figure 129. Creating a datatype property	281
Figure 130. Creating a bridging connector (object property)	282
Figure 131. Creating a semantic mapping (object property)	283
Figure 132. Creating a Domain Ontology concept for the semantic mapping	284
Figure 133. Extending a modelling construct via integration.....	285
Figure 134. Editing a modelling construct.....	286
Figure 135. Excerpt of the method that generates the SPARQL SELECT to retrieve datatype properties.....	287
Figure 136. Editing Datatype Property of a modelling construct	288
Figure 137. Excerpt of the methods that generate SPARQL SELECT to retrieve (1) semantic mappings and (2) bridging connectors.....	289
Figure 138. Functionality Hide Modelling Construct.....	290
Figure 139. Excerpt of the method that generate SPARQL DELETE DATA and INSERT DATA to hide a modelling construct from the palette.....	290
Figure 140. Thrown exception when trying to delete a modelling construct with child elements	291
Figure 141. Evaluation activities within a Design Science Research process (Sonnenberg & vom Brocke, 2012).....	295
Figure 142. Requirement fulfilment overview through the AOAME's functionalities.....	301
Figure 143. Use cases implemented through AOAME's functionalities	306
Figure 144. DSML4PTM process model containing a Discretionary Task in a BPMN Lane	308
Figure 145. Conceptual solution (a) before and (b) after adding Discretionary Task to Manual Task.....	309
Figure 146. Excerpt of the Palette Ontology and Modelling Language Ontology related to BPMN 2.0	310
Figure 147. Populating the palette with graphical notations from the “Process Modelling View” of BPMN 2.0.....	310
Figure 148. Steps to extend “Manual Task” from BPMN with “Discretionary Task” from CMMN.....	311
Figure 149. Instance of SPARQL Rule 1 dynamically generated after adding Discretionary Task.....	311
Figure 150. Query results (a) before and (b) after domain-specific adaptations in Use Case 1: Adding Discretionary Task	312
Figure 151. Process annotation with pre-defined requirements.....	313

Figure 152. Extension of BPMN modelling element "Group" with three (domain-specific) sub-concepts.....	314
Figure 153. Conceptual solution (a) before and (b) after domain-specific adaptations for Extending BPMN Group use case	316
Figure 154. Populating the palette with graphical notations from the “Process Modelling View” of BPaaS.....	318
Figure 155. Steps to extend the BPMN “Group” with BPR annotation “Generating Customer Billing Data”	319
Figure 156. Query results after domain-specific adaptations in Use Case 2: Extending BPMN Group	322
Figure 157. Bridging connector between the ICF Standard document and KoGu Data Object	325
Figure 158. Conceptual solution (a) before and (b) after adding and referring ICF Standard document.....	326
Figure 159. Excerpt of the Palette Ontology, Modelling Language Ontology and Domain Ontology related to the modelling languages DKMM, BPMN and ICF	328
Figure 160. Steps to extend the modelling element “Data Document” with “ICF Standard”	330
Figure 161. Query results after domain-specific adaptations in Use Case 3: Adding and Referring ICF Standard document	332
Figure 162. Conceptual solution for Use Case 4: Deleting ICF Standard document and properties.....	335
Figure 163. Steps to delete the modelling element ICF Standard.....	336
Figure 164. Query results about the properties (a) before and (b) after the deleting the ICF Standard document.....	338
Figure 165. Query results about the modelling element (a) before and (b) after deleting ICF Standard document.....	338
Figure 166. Steps required to delete a bridging connector (i.e. object property) in the GUI	340
Figure 167. Query results (a) before and (b) after deleting the property	341
Figure 168. SPARQL query to show current annotation properties of ICF Standard	343
Figure 169. Steps to change annotation properties of a modelling construct on the GUI.....	344
Figure 170. Result of the dynamically generated SPARQL SELECT that retrieves bridging connectors	345
Figure 171. Steps to change bridging connectors of a modelling construct on the GUI	346
Figure 172. Steps to change semantic mappings of a modelling construct on the GUI	347
Figure 173. Result of the dynamically generated SPARQL SELECT that retrieves datatype properties.....	348
Figure 174. Steps to change datatype properties of a modelling construct on the GUI	349
Figure 175. Steps to hide a modelling construct from the palette	350
Figure 176. Query results (a) before and (b) after hiding a modelling element from the Palette	351
Figure 177. The agile and ontology-aided met-modelling approach to support Design Thinking	353

Figure 178. Main artefact, sub-artefacts and knowledge flow among them..... 362

Figure 179. Adding constraints from the graphical modelling language (left) or from the ontology (right)370

LIST OF TABLES

Table 1. Ontological, epistemological and axiological assumption. Adapted from (Saunders et al., 2019)	64
Table 2. Aspects of the pragmatism. Adapted from (Goldkuhl, 2011).....	65
Table 3. Characteristics of an inductive research approach. Adapted from (Saunders et al., 2019).	66
Table 4. An excerpt of the definition of actors in the research project Patient Radar	108
Table 5. Process requirements - application scenario.....	113
Table 6. Document requirements - application scenario	114
Table 7. Information systems and data requirements - application scenario	114
Table 8. Decision requirements - application scenario.....	114
Table 9. Process requirements - reference process model	118
Table 10. Document requirements - reference process model.....	118
Table 11. Information systems and data requirements - reference process model	119
Table 12. Decision requirements - reference process model	119
Table 13. Additional requirements	120
Table 14. Overview of the covered requirements	127
Table 15. Overview of the needed concepts from CMMN and Control Element Meta-Model	128
Table 16. Overview of the needed concepts from the DMN meta-model.....	131
Table 17. Overview of the needed concepts from the Document and Knowledge Meta-Model	133
Table 18 Overview of the needed concepts from the Organisational Meta-Model.....	134
Table 19. Additional semantics for BPMN modelling constructs and extensions	135
Table 20. An excerpt of the graphical notation of DSML4PTM.....	139
Table 21. New Design Decisions.....	143
Table 22. Fulfilment of document requirements.....	145
Table 23. Comparison of problems identified in the Create phase for DSML4PTM and BPaaS	156
Table 24. Comparison of problems identified in the Design phase for DSML4PTM and BPaaS DSML.....	157
Table 25. Comparison of the problem identified in the Develop phase for DSML4PTM and BPaaS	158
Table 26. Comparison of the problem identified in the Deploy/Validate phase for DSML4PTM and BPaaS	159
Table 27. APQC Top Level Categories	174
Table 28. Non-functional IT Specifications.....	183
Table 29. Mapping requirements to basic functions CRUD.....	207

Table 30. SPARQL Rule 1 – Integrate modelling elements from different modelling languages	234
Table 31 An instance of SPARQL Rule 1	236
Table 32. SPARQL Rule 2 – Create modelling construct	240
Table 33. SPARQL Rule 3 – Create object property.....	241
Table 34. SPARQL Rule 4 – Create datatype property	242
Table 35. SPARQL Rule 5 – Create domain concept.....	243
Table 36. Instance of SPARQL Rule 2.....	245
Table 37. Instance of SPARQL Rule 4.....	246
Table 38. Instance of SPARQL Rule 3b.....	247
Table 39. Instance of SPARQL Rule 3a.....	248
Table 40. Instance of SPARQL Rule 5.....	249
Table 41. SPARQL Rule 6 – Delete modelling construct	251
Table 42. SPARQL Rule 7 – Delete property.....	252
Table 43. Alternative to SPARQL Rule 7	252
Table 44. An instance of SPARQL Rule 6	253
Table 45. An instance of SPARQL Rule 7	253
Table 46. SPARQL Rule 8 – Update modelling construct	255
Table 47. SPARQL Rule 9 – Update object property.....	256
Table 48. SPARQL Rule 10 – Update datatype property	257
Table 49. SPARQL Rule 11 – Update the datatype property to hide a modelling construct	257
Table 50. An instance of SPARQL Rule 8 - update modelling construct	258
Table 51. An instance of SPARQL Rule 9 - update object properties	259
Table 52. An instance of SPARQL Rule 10 – update datatype properties	260
Table 53. An instance of SPARQL 11 - hide a modelling construct.....	260
Table 54. Ex post artificial and naturalistic evaluation strategy for the artefact. Adapted from (Pries-Heje et al. 2008)	296
Table 55. Evaluation criteria for Design Science Research artefact types (March and Smith, 1995)	297
Table 56. Evaluation Strategy for the Agile and Ontology-Aided Meta-modelling Approach	299
Table 57. SPARQL query to prove consistency in Use Case 1: Adding Discretionary Task	312
Table 58 SPARQL query to prove consistency in Use Case 2: Extending BPMN Group....	323
Table 59 SPARQL query to prove consistency in Use Case 3: Adding and Referring ICF Standard Document”	333
Table 60 SPARQL query to prove that properties of ICF Standard document are deleted...	338
Table 61. SPARQL query to prove that ICF Standard document is deleted	339
Table 62. SPARQL query to prove consistency after deleting properties.....	341
Table 63. SPARQL query to retrieve the value of a datatype property.....	351
Table 64. Main artefact contribution	363

Table 65. Sub-artefacts contribution.....	364
Table 66. Adherence of the research to DSR guidelines from Hevner et al. (2004)	368
Table 67 Abbreviation overview	391

APPENDIX A: PATIENT TRANSFERAL MANAGEMENT DOCUMENTATION

The below table contains the sources of the documents related to case Patient Transferal Management. Documents can be found in the folder [Appendix A](#).

Folder	Folder description	File name
A1. Information Model	This folder contains the information model, which describes all the terms and related meaning. It was created in the domain analysis of the project. The folder contains also supportive material to the creation of the information model such as the presentation of the project in PDF and the healthcare standards.	<ul style="list-style-type: none"> - Information Model (source - workshop SwissPost-Hospital Walenstadt).docx - 2012-04-12_PatientenRadar_V5.pdf - ...\\Healthcare Standards\
A2. Conceptual Model	This folder contains the conceptual model, which presents a lightweight ontology describing the domain. Classes, relations and attributes are used in a UML class diagram manner.	<ul style="list-style-type: none"> - ConceptualModel.pdf
A3. Reference Model - Elective Case	This folder contains the last version of the BPMN reference process model of an elective entry case (v11). The process model can be found in different format (jpeg, bpmn) and in a pdf document. It also contains the previous process models that evolves over the course of the project. The folder also contains supportive material to the process such as the information care from one project partner (Reha Klinik Valens) as we as a Power Point with the decision models.	<ul style="list-style-type: none"> - Reference Process Model v11 -Final Version for Requirements Elicitation.jpeg (and in .bpmn) - BPMNReference Process Model for Elective Entry Cases.pdf - ...\\Old versions of the reference process model.zip - Form the Reha Klinik Valens containing information care value range.pdf - DecisionModels and BPMN for Patient Radar.pdf
A4. Application Scenario - Emergency Case	This folder contains the application scenario of an emergency case in a PDF document. The folder also contains a sub-folder with the material produced in a workshop with a transferal manager. The workshop was used as basis to start describes the application scenario.	<ul style="list-style-type: none"> - Application Scenario - A Geriatric Patient with Stroke.pdf - ...\\Material from workshops with Transferal Manager in Hospital of Grab 14-03-2016.zip
A5. Cost Reimbursement KoGu	This folder contains a PDF document describing the data needed for the cost reimbursement form (KoGu). The data can be found between page 31 and page 37 of the document.	<ul style="list-style-type: none"> - Cost Reimbursement KoGu - between 31 and 37.pdf
A6. Mockups for the Reference Process Model	This folder contains the three documents: a PDF document describing the elective reference model and mock-ups, a PDF document containing the list of mock-ups for the reference model, and finally a .zip file containing the interactive mock-ups.	<ul style="list-style-type: none"> - Reference Model and Mock-ups.pdf - Mock-ups TM.pdf - Mock-ups Transferral Management.zip
A7. Reference Process Model Activities	This folder contains the reference process model activities in the form of a mind map. Each activity contains a number that reflects the execution order in the process. For each activity there is a mock-up that corresponds to the same number (see A6).	<ul style="list-style-type: none"> - ...Reference process model activities.bmp
A8. Graphical Notation DSML4PTM	This folder contains the graphical notations for the modelling construct of DSML4PTM. The graphical notations can be found in a PDF document as well as in the GraphRep format, which are importable to the ADOxx modelling tool.	<ul style="list-style-type: none"> - GraphicalNotationDSML4PTM.pdf - GraphRep-20180426T130751Z-001.zip

A9. ADOxx Library for DSML4PTM	This folder contains the DSML4PTM ADOxx Library and Models. The ADOxx library are in the file DSML4PTM.abl and the ADOxx models are in the file models.adl. Both files can be imported to ADOxx for usage.	<ul style="list-style-type: none"> - DSML4PTM.abl - Models.adl
A10. Evaluation Scenarios for DSML4PTM	This folder contains the material used and produced during the evaluation of the DSML4PTM. The folder contains a PDF document with the fulfilment of requirements of DSML4PTM. The folder also contains three sub-folders “KoGu accepted”, KoGu rejected” and “Scenario”. Each of them contains PDFs depicting models. Models in the folder “KoGu accepted” represent the happy path, mainly derived from the reference process model. Models in the folder “KoGu rejected” is a deviation of the reference process model. Finally, models in the third folder “Scenario” reflect the application scenario geriatric patient with stroke.	<ul style="list-style-type: none"> - ...\ DSML4PTM Models\KoGu accepted.zip - ...\ DSML4PTM Models\KoGu rejected.zip - ...\ DSML4PTM Models\Scenario.zip - Fulfillment of Requirements.pdf
A11. Focus Group for DSML4PTM	This folder contains the material used and produced during the focus group for DSML4PTM. Namely, the folder contains the PDF of the questionnaire, the PDF containing two scenarios to be proposed after the DSML4PTM was introduced, the zip folder “Questionnaire results” containing the results of the 5 experts involved in the focus group.	<ul style="list-style-type: none"> - Questionnaire results.zip - Questionnaire.pdf - Workshop with two scenarios.pdf
A12. Semantics of modelling constructs of DSML4PTM	This folder contains the semantics of the modelling constructs of DSML4PTM	<ul style="list-style-type: none"> - Semantics of modelling constructs from DSML4PTM.pdf
A13. Fulfilled Requirements for DSML4PTM	This folder contains three PDF documents describing the fulfilment of the requirements for DSML4PTM.	<ul style="list-style-type: none"> - Fulfilled Process Requirements for DSML4PTM.pdf - Fulfilment of Decision Requirements for DSML4PTM.pdf - Fulfilment of the Additional Requirements for DSML4PTM.pdf

APPENDIX B: BUSINESS PROCESS AS A SERVICE DOCUMENTATION

The below table contains the sources of the documents related to case Business Process as a Service. Documents can be found in the folder [Appendix B](#).

Folder	Folder description	File name
B1. Modelling Framework for BPaaS	<p>This folder contains the deliverable titled Modelling Framework for BPaaS D3.1. The document contains the specification of the hybrid modelling environment of CloudSocket named as BPaaS Design Environment. The latter supports the business and IT alignment in the cloud through the semantic lifting of human interpretable models. For this, the BPaaS Design Environment presents two modelling components: the BPaaS modelling environment (for the human interpretation) and the BPaaS Ontology (for the machine interpretation).</p> <p>The BPaaS Ontology development can be found in Chapter 5 (page 79). The formal description of the BPaaS meta-model can be found in Sub-section 4.2.3 (page 72).</p>	- CloudSocket_D3.1_BPaaS_Design_Environment_Research.pdf
B2. Modelling Prototypes for BPaaS	<p>This folder contains the deliverable titled Explanatory Notes: Modelling Prototypes for BPaaS D3.2. The document builds on the document in folder B1. It contains the further development of (a) the modelling language BPaaS (i.e. requirements and specifications) implemented in the ADOxx platform, (b) the BPaaS Ontology, (c) semantic rules for the semantic matchmaking between business processes and workflows/cloud service specifications, (d) the implementation, setup and configuration of the BPaaS Design Environment prototype.</p> <p>See the complete list of the new attributes after the first engineering lifecycle iteration in Section 3.3.</p> <p>The complete list of non-functional business process requirements can be found in Section 3.3.1.2.</p> <p>The complete list of non-functional workflow descriptions can be found in Section 3.3.2.2</p>	- CloudSocket_D3.2_Modelling_Prototypes_for_BPaaS.pdf
B3. BPaaS ADOxx Library and Models	<p>This folder contains the BPaaS ADOxx Library and Models. Namely, there are (1) the latest version of the BPaaS Modelling Library developed in ADOxx with extension .abl (i.e. BPaaS meta-model and graphical notations); (2) the old versions of the BPaaS Modelling Library developed in ADOxx with extension .abl (i.e. BPaaS meta-model and graphical notations); (3) the ADOxx models for the three scenarios Christmas card, Social Media Campaign and Send Invoice. Each of them has a dedicated folder containing the ADOxx files with extension.adl as</p>	<ul style="list-style-type: none"> - Latest version of the BPaaS modelling library retrievable at: - BPaaS Modelling Library – Final Version\BPaaS Modelling Library – Final Version.abl - Old versions of the BPaaS modelling library: - ...\Old Libraries - Christmas Card Models: - ...\Christmas Card Models - Social Media Campaign Models: - ...\Social Media Campaign Models

	well as images for the convenience of the reader. Also, each model folder contains the correspondent library.	<ul style="list-style-type: none"> - Send Invoice Models: - ...\Send Invoice Models
B4. Knowledge Provision for CloudSocket Findings and Feedback	This folder contains the deliverable titled Knowledge Provision for CloudSocket Findings and Feedback D6.2. The document shows the progress of all research and technical items made during the CloudSocket research project. The progress is presented in the form of description of the latest improved items and the motivation for their improvements. The latter takes the name of “lessons learned” in the document. Only Section 2.1 and 2.2 are regarded as relevant for this research work.	<ul style="list-style-type: none"> - CloudSocket_D6.2_Knowledge_Provision_for_CloudSocket.pdf
B5. Image of Models	This folder contains the images of models. The images represent the models business processes and workflows models of two use case scenarios: Send Invoice and Social Media Campaign. Images were generated with the ADOxx Modelling Toolkit.	<ul style="list-style-type: none"> - ...\Send Invoice\ - -Business Process Send Invoice.bmp - -Workflow - Ninja+YMENS.bmp - ...\Social Media Campaign - -BP-Social Media Campaign-Basic.png - -Wf-Social Media Campaing - Basic.png
B6. BPMN Send Invoice and Workflow	This folder contains a PDF document containing BPMN Send Invoice and a workflow implemented in business processes.	<ul style="list-style-type: none"> - Send Invoice BP and a WF.pdf
B7. Business-like questions to retrieve Cloud solutions	This folder contains a master’s thesis describing the business-like questions to retrieve Cloud solutions. The description is based on an interview with project partner Mathema.	<ul style="list-style-type: none"> - Towards a Semantic Enrichment of Business-IT Alignment in the Cloud.pdf
B8. Non-Functional Cloud Service Specifications for BPaaS	This folder contains a PDF document with all the possible values for non-functional cloud service specifications for BPaaS. The specifications are grouped by sub-dimensions and dimensions.	<ul style="list-style-type: none"> - Non-Functional Cloud Service Specifications for BPaaS.pdf

APPENDIX C: MODELLING EXPERT INTERVIEWS DOCUMENTATION

The below table contains the sources of the documents related to modelling expert interviews. Documents can be found in the folder [Appendix C](#).

Folder	Folder description	File name
C1. Expert Interview - enterprise architect	This folder contains the pdf document with the enterprise architect as well as a model built with an ArchiMate extension.	- Expert Interview - enterprise architect.pdf - ModelBuiltwithArchiMateExtension.pdf
C2. Expert Interview - enterprise modeller	This folder contains the pdf document with the interview results of the business process modeller as well as a scenario describing a meta-model extension (GDPR scenario).	- Expert Interview - process modeller.pdf - Implementation of DSGVO (GDPR) - Impact CH.pdf - Meta-Model implemented in Adonis – GDPR extension.pdf
C3. Expert Interview - process modeller	This folder contains the pdf document with the interview results of the workflow modeller. The folder also contains the model and scenario where BPMN has to be extended (Dynamic Batch Scenario).	- Expert Interview - workflow modeller.pdf - Camunda models – dynamic batch scenario.pdf
C4. Expert Interview - workflow modeller	This folder contains the pdf document with the interview results of the enterprise modeller.	- Expert Interview - enterprise modeller.pdf
C5. Questionnaire Template for Experts Interview	This folder contains a PDF document with the template of the Interview.	- Questionnaire Template.pdf

APPENDIX D: VALIDATION SPARQL RULES DOCUMENTATION

The below table contains the sources of the two documents describing the syntactic and semantic validation activities of the 11 semantic rules. Documents can be found in the folder [Appendix D](#).

Folder	Folder description	File name
D1. Syntactic Validation of SPARQL Rules	This folder contains a PDF document describing the syntactic validation of the 11 SPARQL Rules. The validation took place through the SPARQL Update Validator.	- Syntactic Validation of SPARQL Rules.pdf
D2. Semantic Validation of SPARQL Rules	This folder contains a PDF document describing the semantic validation of the 11 SPARQL Rules. The validation took place by instantiating the SPARQL rules to fit use cases from the Patient Transferal Management. The SPARQL rule instances are fired against the DSML4PTM ontology in TopBraid.	- Semantic Validation of SPARQL Rules.pdf

APPENDIX E: PROTOTYPE DOCUMENTATION

The below table contains the sources of the files and documents of the prototype AOAME. The documentation can be found in the folder [Appendix E](#).

Folder	Folder description	File name
E1. SPARQL query samples	This folder contains a PDF document describing four SPARQL query samples. The queries are dynamically generated for retrieving knowledge to display the graphical notations in the Palette.	- SPARQL query samples dynamically generated for the Palette.pdf
E2. Angular WebApp	This folder contains the zip file with the Angular Web App.	- OntologyBasedModellingEnvironment-WebAppNew.zip
E3. Web Service	This folder contains the zip folder with the Java-based Web Service.	- OntologyBasedModellingEnvironment-WebService.zip
E4. Ontologies for the Modelling Environment	This folder contains the zip file with the ontologies for AOAME. The ontologies that were used, developed and extended during the prototype implementation and evaluation.	- Ontology4ModellingEnvironment.zip
E5. Graphical notations for AOAME5	This folder contains the zip file with the graphical notations developed for the implementation of use cases.	- images.zip
E6. Getting Started with AOAME - Guidelines	This folder contains a PDF document with the getting started guide of AOAME. Guidelines are provided for using AOAME both locally and online. The folder also contains Apache Jena Fuseki to start AOAME locally.	- Getting Started Guide of AOAME.pdf - apache-jena-fuseki-3.4.0.zip

APPENDIX F: EVALUATION DOCUMENTATION

The below table contains the sources of the documentation related to the evaluation of the agile and ontology-aided meta-modelling approach. The documentation can be found in the folder [Appendix F](#).

Folder	Folder description	File name
F1. Query Results Use Cases 1 and 2	This folder contains additional screenshots produced during the evaluation of the artefact. The screenshots show the retrieved knowledge before and after the domain-specific adaptations of modelling languages.	<ul style="list-style-type: none">- UseCase2_After performing the modeling language extension.png- UseCase2_Before performing the modeling language extension.png- Usecase3_After performing the modeling language extension.png- UseCase3_Before performing the modeling language extension.png
F2. 40 retrieved entries	This folder contains the list of retrieved entries from the query that was fired before deleting a modelling construct.	<ul style="list-style-type: none">- queryResults_BeforeDeletingModellingConstruct.csv