

Expert Interview

First Part

When: 05/01/2018

Where: Zurich

Expert: Felix Brandmayr

Role: Enterprise Architecture and Business Process Management Senior Consultant

Company: BOC Group

Company Website: <https://ch.boc-group.com/>

Personal Profile: <https://www.linkedin.com/in/brandmayr/>

In Enterprise Architecture consulting projects:

1. *Did users face difficulties in learning standard modeling languages and using them in ADOit?*

Most clients face difficulties in learning and modeling with standard modeling languages. In enterprise architecture consulting projects, some clients choose to stick to a standard like ArchiMate or Togaf, weather others prefer to adopt a (non-standard) modeling language and mechanisms that are based on BOC projects experience. The latter include aspects of modeling standards too. Both approaches can be supported by our professional modeling tool ADOit.

- a. *If yes, what are the reasons for that?*

Standard modeling languages are usually too complex for clients. This is mainly for three reasons:

- (1) standard modeling languages cover too many aspects that are not needed in a specific project.
- (2) the mismatch between the level of abstraction of the modeling standard and the client expertise. For example, in the case of ArchiMate, it results to be rather technical for business people to understand.
- (3) standard languages embed semantics to which clients are not familiar with. In other words, the meaning of many modeling constructs (i.e. both modeling elements and modeling relations) is unknown.

- b. *How could this issue be overcome?*

The three above issues are addressed as follows, respectively:

- (1) ADOit allows adapting the metamodel such that a simplification of the language performed.
- (2) Training and documentation (including vocabulary)
- (3) Training and documentation (including vocabulary)

In both research and industry, there is the recent trend of adapting standards (or existing) modeling languages to address a specific domain. In result a domain-specific modeling language is developed (DSML). This has the following benefits:

- It decreases the error prone while modeling. This is due the injection of semantics (i.e. abstract syntax and constraints) in the metamodel, which decreases the degree of freedom of modelers.
- It enhances understanding of models by domain experts. This is due to the graphical notations targeting a specific domain.

Developing a DSML through domain-specific adaptation of existing modeling languages has the following benefits:

- Modeling expert-friendly. This is due to the reference to already existing modeling standards.
- Total or partial reusability of the resulting language (i.e. DSML) within the modeling community. This is also due to the reference to already existing modeling standards.
- It fosters the quality of the modeling language. Established experience and lessons learned from existing modeling languages can be taken into account. Additionally, semantics and syntax can be borrowed.

2. *Could a DSML address one of the issues/problems identified in question 1? Are there more problems that can be addressed?*

DSMLs have the potential to address all the three issues identified in answer 1. If we consider the result of a simplification of a modeling language as a DSML, than it is often the case that in projects a DSML is needed.

3. *Have there been situations, where the modeling languages were not sufficient and where an adaptation could have made sense? Namely, adapting language constructs to fit a more specific domain?*

Yes, as above answered, it is often the case in enterprise architecture consulting projects, where an adaptation of the modeling language needs to be performed. If clients want to stick to the standard, the modeling language adaptation is mainly restricted to a simplification of the modeling language. This means making modeling constructs invisible from the palette. Also, adaptation in terms of restrictions on attributes (which to consider and which not) and attribute values is likely to occur. In case a client does not need to stick to a standard modeling, the adaptation is also performed on the graphical notation to customize modeling constructs. The latter is however not a common practice by IT Architects as it is not regarded as top priority. Therefore it really depends which target of stakeholders is addressed by the language. Specification of modeling constructs (i.e. creating sub classes). The adaptation of a modeling language is performed in ADOit (in the metamodel). For instance, when adapting the standard Archimate, ADOit implements the «profiling» specialization mechanism offered by the standard. Namely, sets of attribute types can be specified and a predefined value can be assigned to each of these attributes. These data structure can be coupled with elements or relations.

Specialization of modeling elements and relations also can occur and the reason is mainly to perform restrictions on attributes.

a. *If not, Why?* Not applicable.

b. *Could a functionality for an on-the-fly customization of modeling constructs be useful in projects with domain-specific target?*

Yes, see above.

4. For the situations, where the language was not adequate, what kind of modifications on the modeling language would have helped? e.g. creating/deleting/update a modeling construct (i.e. class, attribute and relation on the metamodel level)

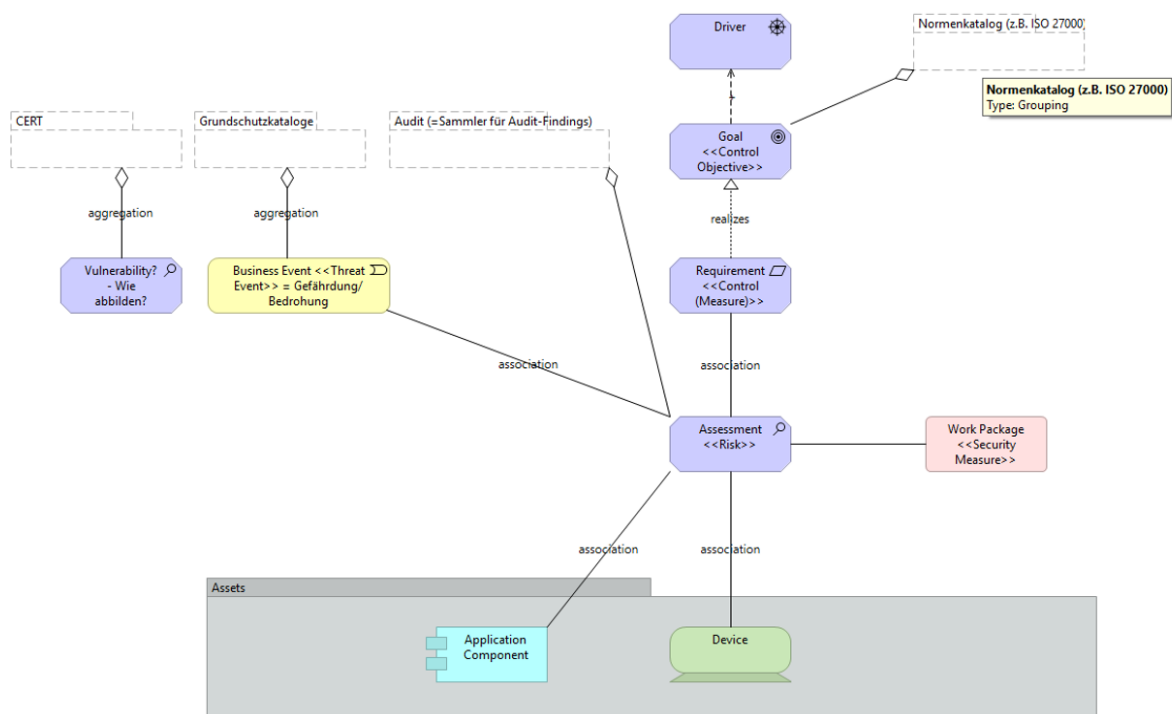
As already mentioned, most of the modeling language adaptation occur in terms of attribute restrictions. Also new attributes can be entered. Therefore, updating and creating new attributes. In terms of graphical notation the most common customization is by assigning colors, sometime by changing the shape and other times by providing visibility to the attributes directly in the graphical notation, e.g. lifecycle date of the modeling constructs.

The creation of subclasses, if occurs, in ADOit is performed through the extensibility mechanism stereotype. Update can be performed to change a name of a modeling construct.

Delete is never applied for two reasons: (1) it can create inconsistency with respect to both the already existing elements and relations in the metamodel and in the existing models that have already been modelled. (2) It might be useful to use a modeling construct later on in the project. Therefore, we make modeling constructs and do not delete them.

5. Could you provide at least a use case where domain-modeling adaptation would have made sense?

ArchiMate Metamodel extension for ISMS (Information Security Management System) scenarios



Comments:

To quickly address stakeholder concerns, Enterprise architects may have to adapt modeling languages as fast as possible. Therefore, an approach that allows quickly developing consistent metamodels could definitely help to deliver value to the different IT and business stakeholders.

Second Part

1. Do the operators derived in the paper (see below) make sense for you?

Yes, but I discourage to use operators where delete actions occur on modeling elements, relations and on attributes.

Use the delete operators only on the new created elements.

2. Do you suggest other operators/actions on the language for adaptation purpose?

Hide instead of delete.

Operators:

Operator 1: Create sub-class. This operator is applied on modeling elements and modeling relations to create new modeling elements and new modeling relations. This operator is also applied to integrate modeling elements (classes) from different modeling languages. For example, the operator would allow to connect "Discretionary Task" from CMMN as a subclass of the "User Task" from BPMN.

Operator 2: Delete sub-class. This operator is applied on modeling elements and modeling relations to remove unneeded modeling elements and modeling relations from the modeling language.

Operator 3: Create relation (object properties). This operator connects modeling elements and modeling relations to the related Domain Ontology concept.

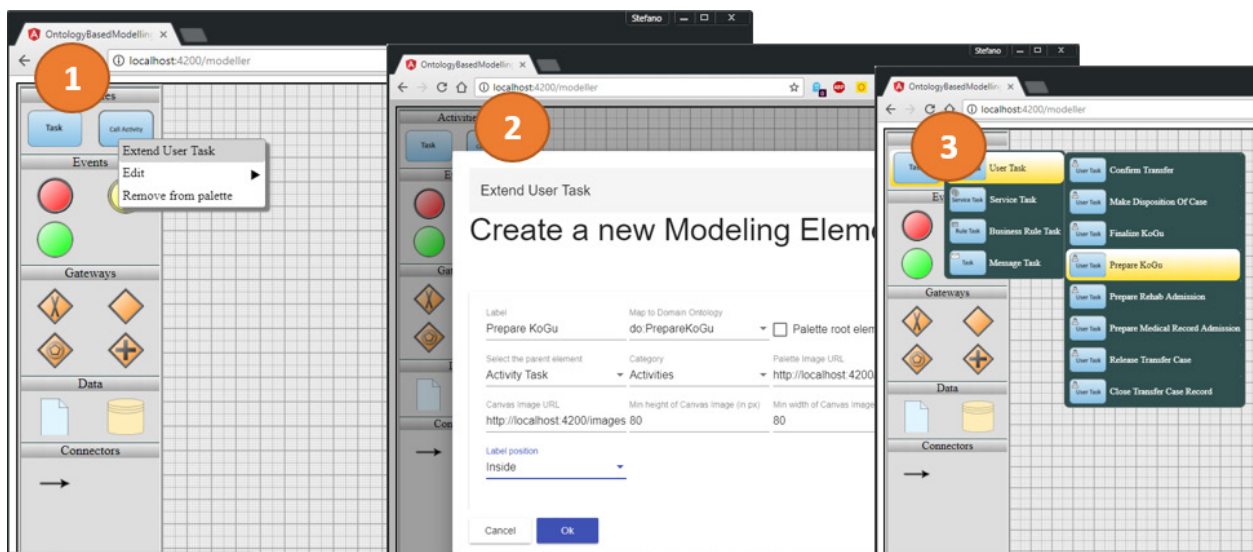


Figure 1. Operator from 1 to 3

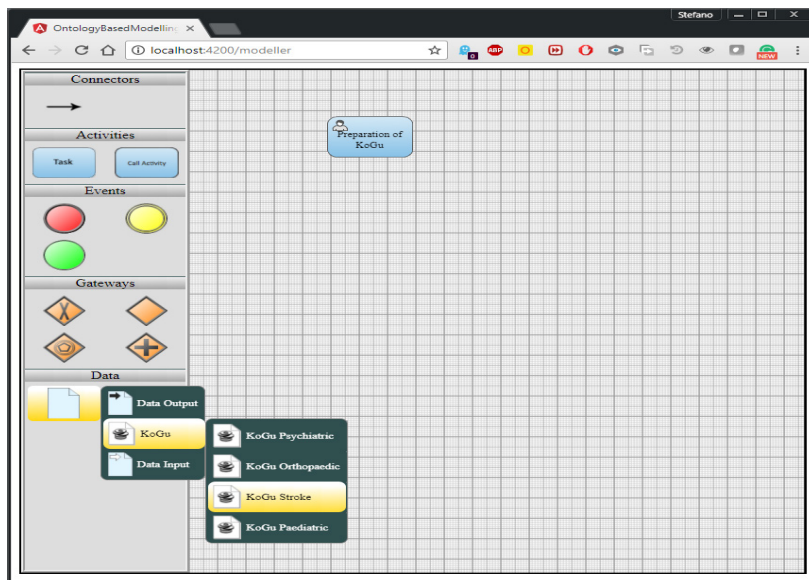


Figure 2. Operator 1 applied on "Data Object"

Operator 4: Update relation (object properties). This operator is applied on as it allows updating existing connections between modeling elements/relations and the related Domain Ontology concepts.

Operator 5: Delete relation (object properties). This operator allows deleting existing connections between modeling elements/relations and the related Domain Ontology concepts.

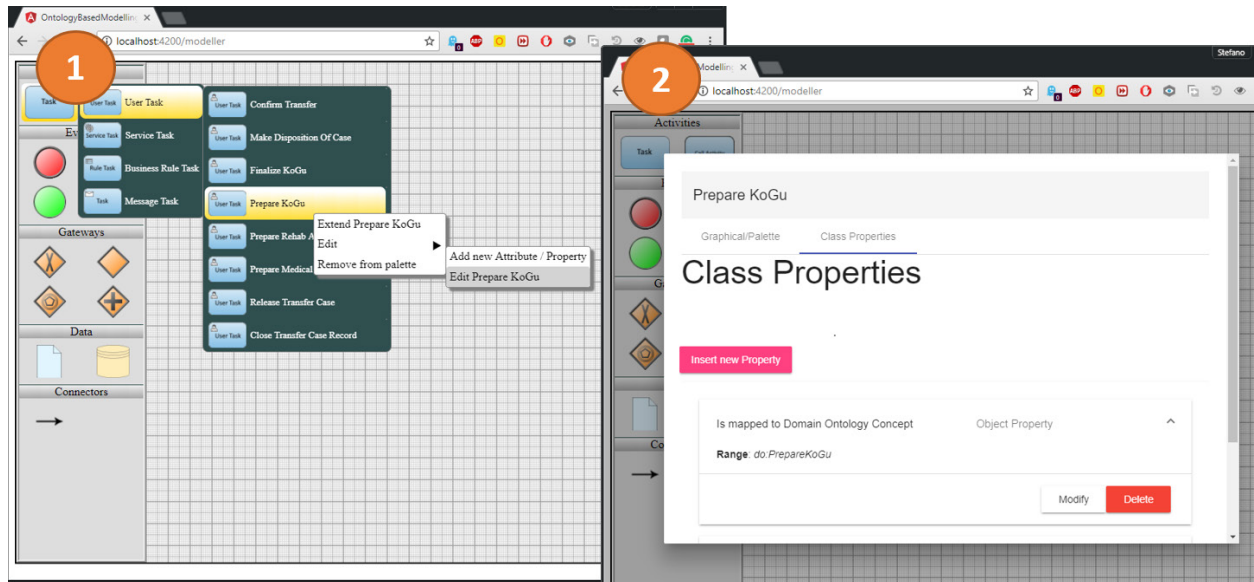


Figure 3. Operators 4 and 5

Operator 6: Create attribute (datatype properties). This operator allows adding new attributes to modeling elements and modeling relations.

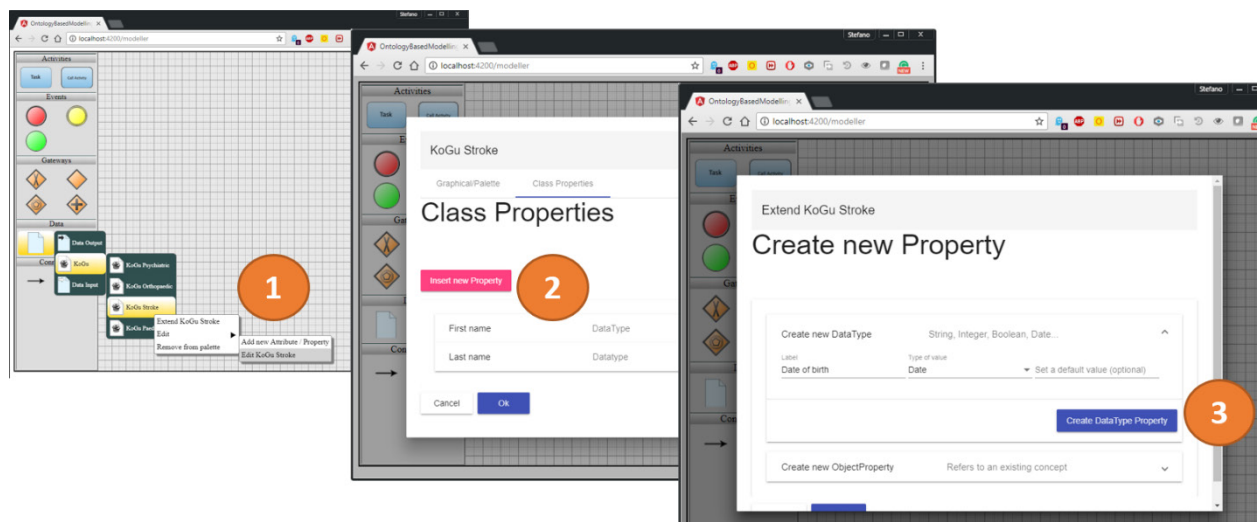


Figure 4. Operator 6 (and 3)

Operator 7: Update attribute (datatype properties). This operator is allows updating existing attributes.

Operator 8: Delete attribute (datatype properties). This operator is allows deleting existing attributes.

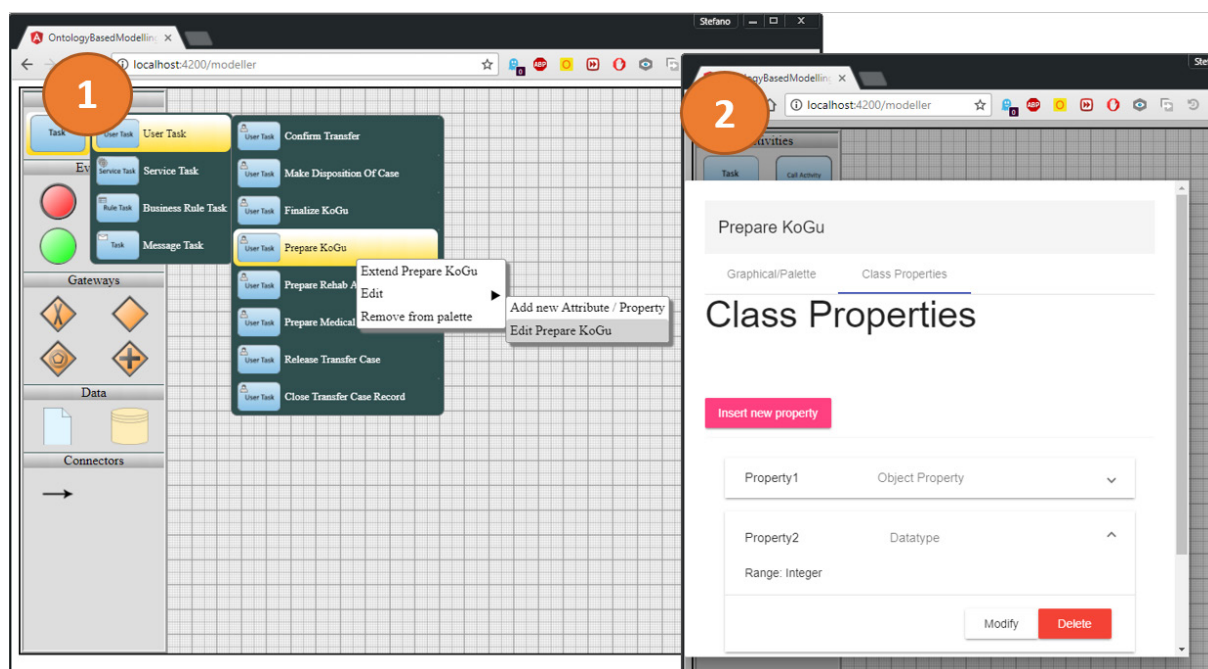


Figure 5. Operatros 7 and 8

Operator 9: Assign attribute type. This operator allows assigning value types String, Integer, Boolean to attributes of modeling elements.

Operator 10: Update attribute types. This operator allows updating types that are assigned to attributes of modeling elements.

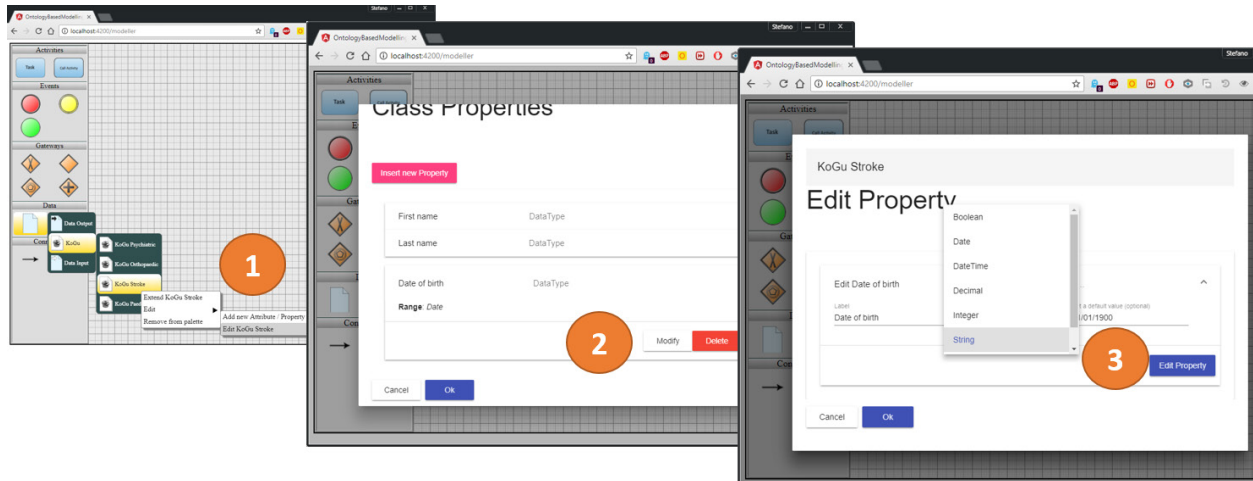


Figure 6. Operator 9 and 10

Operator 11: Enable representation level. This operator allows specifying whether a modeling element is a type or instance. Hence, the modeling environment can model instances as well as classes. Since an class can be an instance of another class, the modeling environment enables to distinguish between different levels of abstractions and not restricted to the class-instance dichotomy of description logics representation.

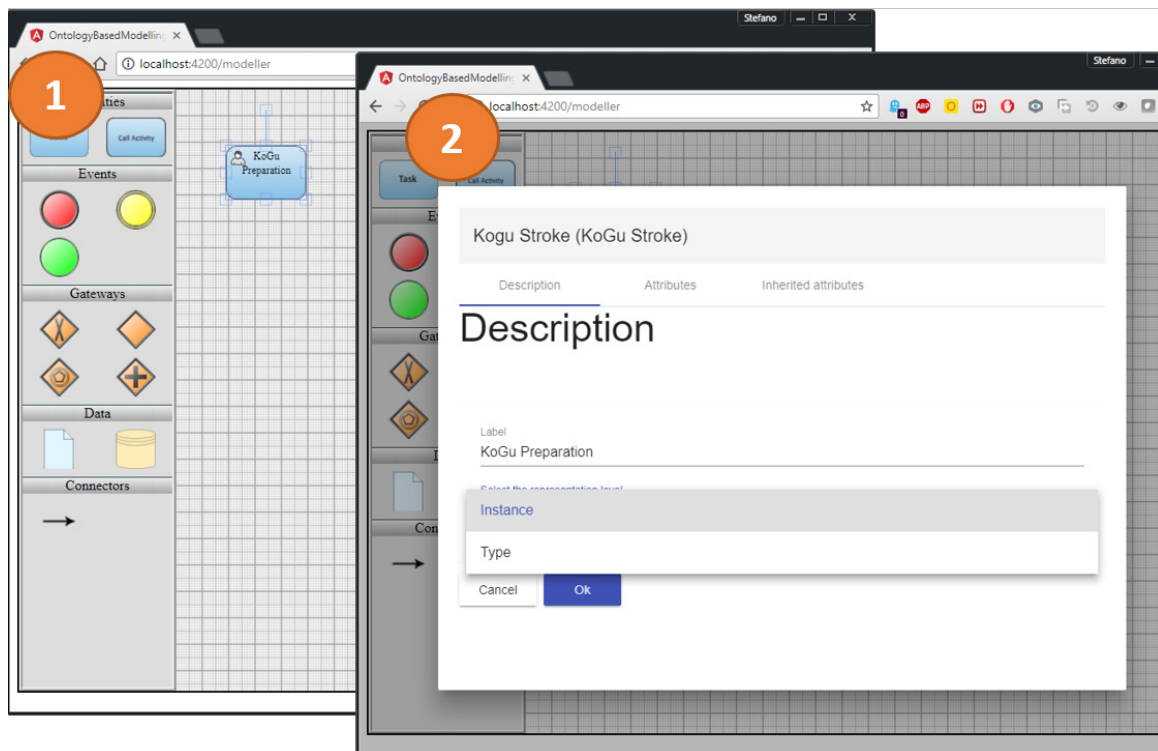


Figure 7. Operator 11

Figure 8 shows an excerpt of the ontology that is behind the modeling environment. It includes elements that belong to the architecture of the ontology-aided modeling environment, i.e. all elements with the “lo” prefix. Elements with the prefix “bpmn” belong to the modeling standard BPMN whereas “dslm4ptm” belong to elements of the created DSML that covers the patient transferal management domain. Finally, the element with the prefix “do” reflects the root concept of the domain ontology, which provides the (language-independent) semantics to both the modeling elements and modeling relations.

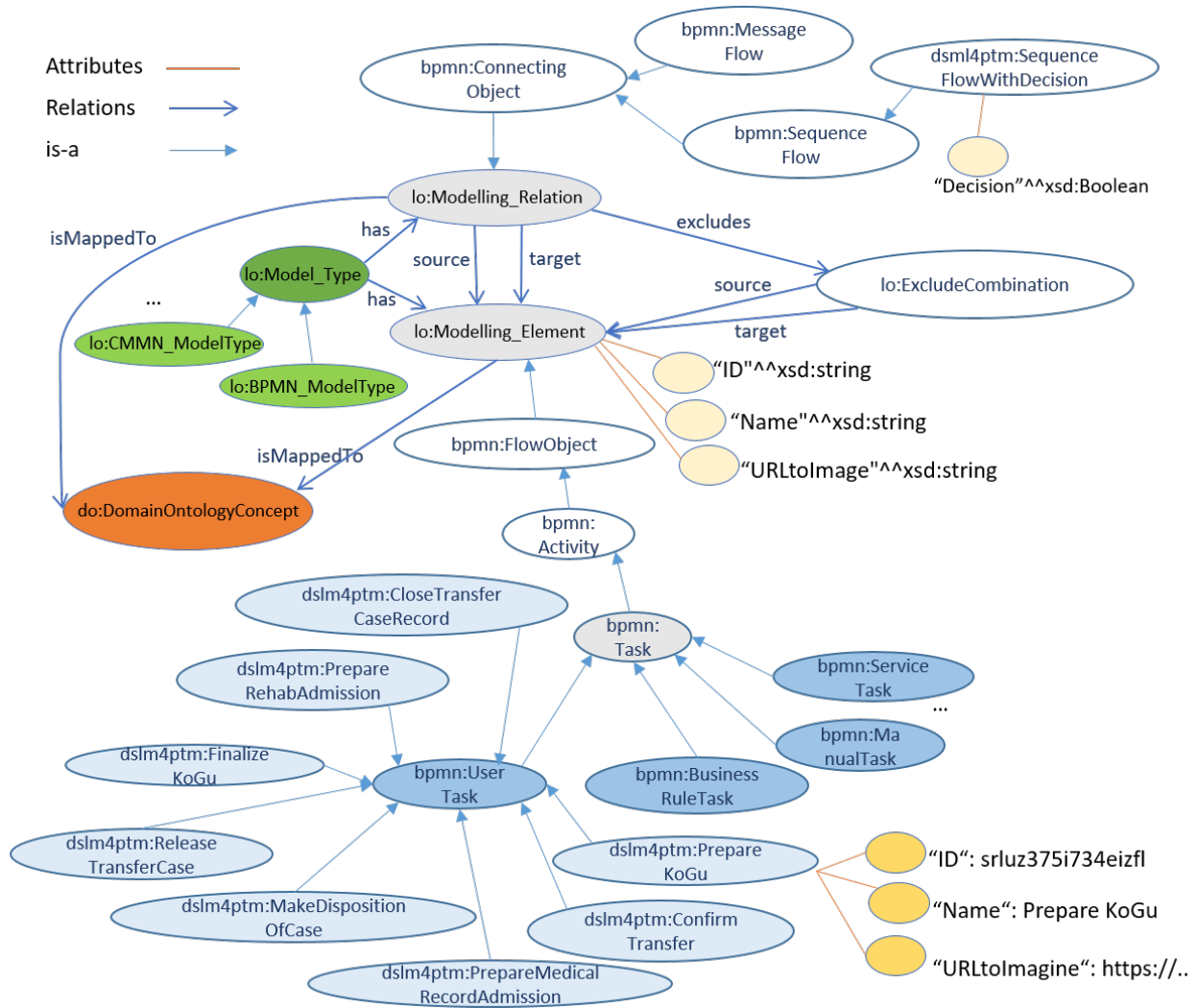


Figure 8. Ontology Excerpt

Figure 9 shows the the overall new approach agile and ontology-aided approach.

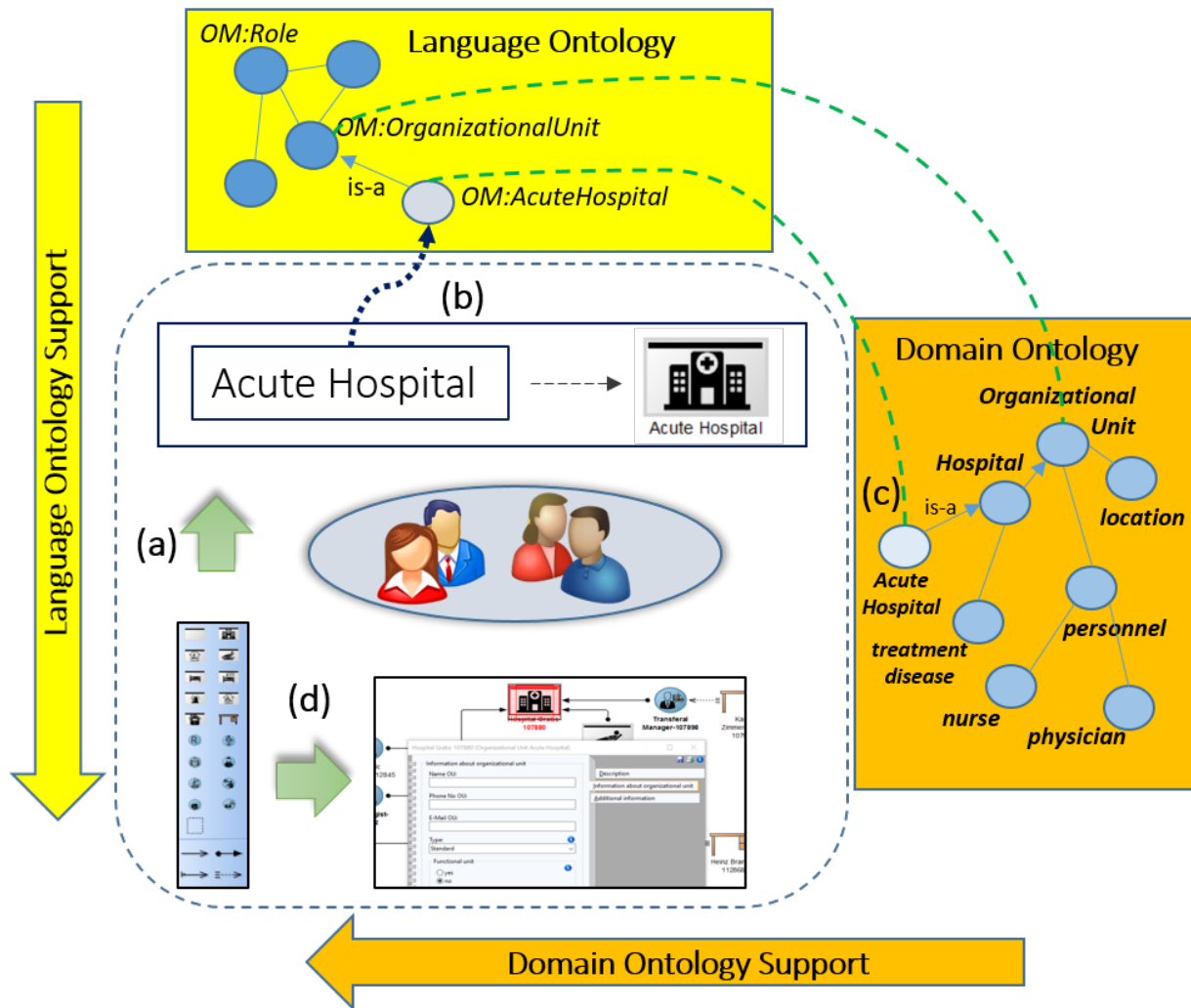


Figure 9. The Agile and Ontology-aided Modeling Environment