Towards an integrative modelling technique between business and information system development

by

Pieter Joubert

Submitted in fulfilment of the requirements for the degree

Philosophiae Doctor (Information Technology)

In the Faculty of Engineering, Built Environment and Information Technology

at the

University of Pretoria

Pretoria, South Africa 2012

Abstract

Candidate: Pieter Joubert
Promoters: Prof C de Villiers
Prof JH Kroeze
Department: Informatics
Degree: Philosophiae Doctor (Information Technology)
Keywords: Information systems development, modelling, modelling techniques, grounded theory, design science research

There are many situations during information system development (ISD) where there is a need to do modelling on a business level before more detailed and robust modelling are done on the technical system level. Most business level modelling uses some form of natural language constructs which are, on the one hand, easy to use by untrained users, but which are too vague and ambiguous to be used in subsequent systems level modelling by systems analysts, on the other hand. The goal of this study is to develop an integrative modelling technique that is easy enough to be used by most business users with little training, but robust and structured enough to be used in subsequent ISD modelling. The term "integrative" in the title refers to the fact that this technique attempts to bridge the current gap between modelling on a business level and modelling on a technical level.

*The research consists of two major phases. During the first phase, an integrative modelling technique is developed using a grounded approach. The data that is used for analysis is a representative example of the major ISD modelling techniques used currently. For instance, to represent all the UML techniques, the **UML 2 standard** is used. The purpose of this first phase is to understand what the fundamental concepts and relationships in ISD are and to develop an integrative technique based on that.*

During the second phase, the resultant artefact created by the first phase is evaluated and improved using the design science research approach. This artefact is used in a

representative set of business modelling situations to evaluate its applicability and suitability as an integrative modelling technique between business and ISD.

The integrative modelling technique is evaluated from three perspectives: how it represents business rules, how it handled various aspects of ISD and how it represents requirements expressed as use cases. These evaluations used the two main design criteria of ease of use for users and at the same time adequate levels of expressive power so that the model can be easily translated into existing ISD modelling languages.

The integrative modelling technique developed identified the following three levels of modelling entities and their relationships:

- *Base entities (corresponding to the morphological level in linguistics)*
- *Structure entities (corresponding to the syntactical level in linguistics)*
- *Role entities (corresponding to the semantic level in linguistics)*

The contribution of this research is to provide a better understanding of the fundamental entities in business and ISD modelling and their relationships in order to improve informal, mostly textual, business modelling.

I declare that

"Towards an integrative modelling technique between business and information system development" is my own work and that all the sources that I have used or quoted have been indicated and acknowledged by means of complete references.

P Joubert

Acknowledgements

To my supervisors, Prof Carina de Villiers and Prof Jan Kroeze, for their unending patience with my many “wanderings in the desert”. Without your guidance and support, this would have been impossible.

To my wife, Elsabe, for never ever complaining about all the many hours not spent with her and for always creating an environment conducive to study.

To God for showing me who I really am – the good, the bad and the ugly.

Overview of table of contents

Table of contents

List of figures

List of tables

Acronyms

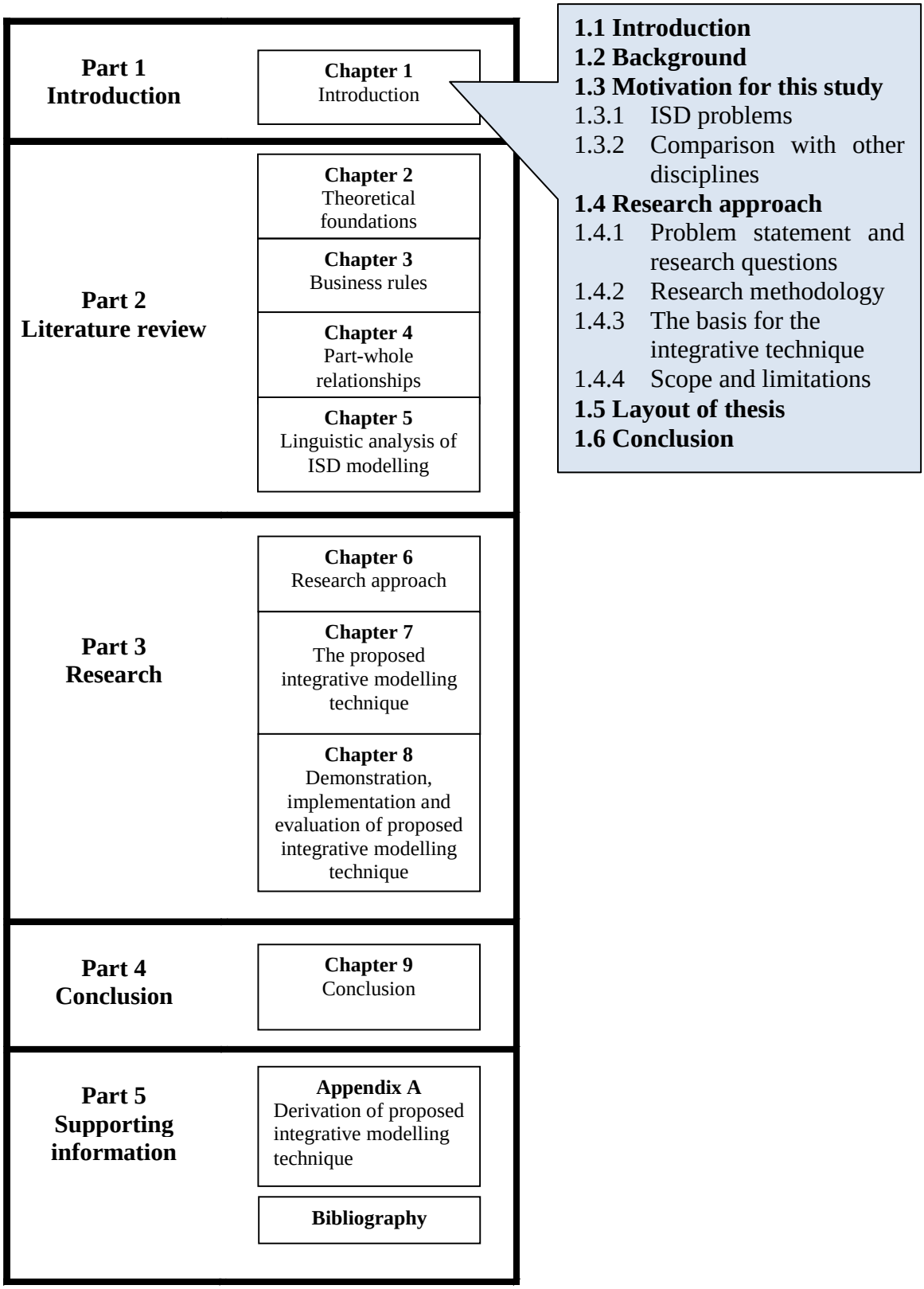
ALC/OLC	Agent/object life cycles
AOR	Agent-object relationship
ARIS	Architecture of integrated information systems
ARM	Agent relationship modelling
BOM	Bill of Material
BPMN	Business Process Modelling Notation
BRM	Business rule management
CASE	Computer-aided software engineering
CAQDAS	Computer-assisted qualitative data analysis software
COTS	Commercial off-the-shelf (products)
CPM	Critical path method
CRUD	Create, read, update and delete
DFD	Data flow diagram
DSRM	Design science research methodology
ECA	Event, condition and action
EOF	End of file
ERA	External receiver actor
ERD	Entity-relationship diagram
EU	European Union
GERAM	Generalised enterprise reference architecture and methodology
GST	General systems theory
GUI	Graphical user interface
HCI	Human-computer interaction
ICEIMT	International Conference on Enterprise Integration Modelling Technology
IDEAS	Interoperability Development for Enterprise Applications and Software
IDEF0	Integration Definition for Function Modelling
IDEF1	Information Modelling Methodology
IDEF1X	Integration Definition for Information Modelling
IDEF3	Process Description Capture Method

IDEF5	Integrated Definition for Ontology Description Capture Method
IEM	Integrated enterprise modelling
I/O	Input-output (interface)
IS	Information system
ISD	Information system development
JAD	Joint application design
JRP	Joint requirements planning
KB	Knowledge base
LAN	Local-area network
LAP	Language action perspective
NoE	Network of excellence
ODP	Open distributed processing
OO	Object orientation
PDA	Personal digital assistant
PERT	Program evaluation and review technique
RAD	Role activity diagrams
RUP	Rational unified process
SDLC	Systems development life cycle
SQL	Structured Query Language
SSADM	Structured systems analysis and design method
SSM	Soft System Methodology
TOGAF	The Open Group Architecture Framework
UEML	Unified Enterprise Modelling Language
UML	Unified Modelling Language
WAN	Wide-area network

Part 1

Introduction

1. Introduction



1.1 Introduction

The researcher, in his work experience, has found that one of the problems hampering information systems development (ISD) is the lack of a truly **integrated** modelling technique or set of techniques. Truly integrated modelling techniques will support modelling during all the phases of the ISD life cycle, from business analysis and systems design to development, and ultimately the maintenance of the resultant system. Integration does not pose such a huge problem during the later phases of system design and development. Techniques such as the Unified Modelling Language (UML) provide for modelling integration during the later phases of the ISD life cycle more than adequately.

The problem investigated in this study is specifically the integration between modelling for business and modelling for ISD. For example, how can we model business rules for an organisation so that business users can easily understand and use it, while at the same time that model has enough expressive power to create a design enabling programmers to implement those same business rules in an information system?

The integration problem has another side to it. Not only is there an integration problem between the business side and ISD, but also between different aspects of business modelling. Zur, Muehlen and Indulska (2010:39) refer to research that point to representational weaknesses in process modelling languages. They speculate that business rule modelling languages can overcome these weaknesses, but the integration of rule and process modelling is seen as problematic. They refer to a recent study showing that organisations frequently supplement their business process modelling notation (BPMN) models with business rules in textual annotations. Similarly, Recker (2010) found in a study on the use of BPMN that a major problem is support in articulating business rules.

Excellent modelling languages and techniques exist for ISD, but very few of them can be simply applied to business modelling. Wilcox and Gurau (2003) identify a number of problems with integration definition for function modelling (IDEF) for

business modelling and then propose UML instead, but then with the provision that it must be extended with an extension like the Eriksson-Penker Business Extensions. The problems are mostly related to complexity, with most business users finding it too difficult to express their business needs in an ISD language or technique, or expressiveness, with users not being able to model everything they require (Recker, 2010). In practice, most business modelling is done in unstructured text format, which leads to unclear, ambiguous descriptions.

Business and ISD modelling is (like architecture, business, education, law and medicine) fundamentally concerned with design and the new, but fast-growing, research approach of design science research as used in this research (Purao et al., 2008). The purpose of this study is to take a step in the direction of creating an integrative modelling technique that bridges the gap between business and ISD modelling. It is done by a two-step process:

- The first step is to develop a **grounded** understanding of the fundamental constructs of business and ISD modelling, trying to answer the question: What must be modelled in business and ISD?
- The second step is to evaluate and improve the integrative modelling technique that was developed in the first step, using **design science research**.

1.2 Background

Iivari, Hirscheim and Klein (2001b) point to the emphasis in ISD literature on improving ISD by the use of new tools, techniques, methods or methodologies. As a result of this emphasis, there is a proliferation of ISD paradigms, approaches, methodologies, methods, techniques and tools. This causes many problems for researchers and practitioners alike. If there are so many tools, techniques, paradigms and approaches, which ones should be used and how can decisions be made about them (Wand and Wang, 1996; Wyssusek, 2006)?

More specifically, Wyssusek (2006:64) considers conceptual modelling and models to be "… among the most fundamental means" in ISD. They developed out of the

need of artificial intelligence to represent knowledge; systems analysis and database development to represent information system problems independent of specific implementations thereof; and programming languages to be more human-centred than technology-centred.

Conceptual ISD modelling is done by using various techniques, ranging from object-oriented techniques like UML diagrams, holistic techniques like soft system methodology (SSM)-rich pictures, data techniques like entity-relationship diagrams (ERDs); process techniques like IDEF0 and BPMN; and many more. These techniques have many commonalities between them, but also many differences. Some of the differences are more obvious, such as concepts existing in one technique but not in another. Other differences are more subtle, such as the concept being called a different name in different techniques or represented differently in different techniques.

What is obvious is that new ISD modelling techniques were developed over the years pragmatically to solve some specific problem area. ERD and normalisation were developed to formalise the modelling of data and databases to overcome the problems faced with databases at that time (such as creating, updating and deleting anomalies) (Chen, 1976). A good current example is BPMN, which was developed to model processes in an organisation. It was developed to overcome the lack of standardisation in current process modelling techniques (Recker, 2010).

In practice, one of the main problems experienced by practitioners and students of business and ISD modelling is how to integrate all of the various ISD modelling techniques. When using more than one of these techniques in the same context, how, for example, do the corresponding concepts of *agent* in agent-object relationship (AOR), *external agent* in data flow diagrams (DFD) and *actor* in use case diagrams relate to each other? And even more confusing, how do the corresponding concepts of *use case* and *system* in use case modelling, *behaviour* in class diagrams, *process* in DFD, IDEF0 and BPMN, *transformation* in SSM, *activity* in activity diagrams; *how* in Zachman, and many other action-oriented concepts relate?

This need for integration exists in a number of dimensions. Firstly, it is needed to integrate modelling techniques across phases. Currently, most ISD projects will start with some form of use case modelling (or a textual requirements specification) to do the requirements analysis. Moving from that to system analysis and design modelling using either UML, ERD, DFD and/or any other modelling diagrams is not a trivial exercise. There is a big jump from the unstructured text in use case narratives and textual specifications to the detail needed in a technical level class or entity diagram. For instance, in a study on practical experiences with eliciting classes from use case descriptions, Cox and Phalp (2007:1286) concluded the following: "It can be construed that there is a lack of detailed guidance about moving from a use case description to elements of design."

Secondly, there is a need to integrate between different areas of modelling (Shen et al., 2004). For instance, the link between process modelling and systems modelling is not well defined. Similarly, modelling the organisation in enterprise architecture and seamlessly integrating that with all other models is problematic (see discussion on Unified Enterprise Modelling Language (UEML) in section 1.3.1).

Thirdly, there is a need to integrate the techniques between non-technical business use (e.g. to define business rules and to define enterprise architectures) and technical IT use (e.g. specifying systems interfaces and designing information system applications).

The purpose of this research is to make a contribution towards the integration of business and ISD modelling techniques. To achieve this, an integrated model for business and ISD modelling that provides integration across phases, between areas and between usages will be developed (by analysing current business and ISD modelling practices as embodied in typical ISD modelling techniques). The development and evaluation of this model will be used to contribute to a better understanding of how ISD techniques can be integrated with business modelling techniques.

1.3 Motivation for this study

1.3.1 ISD problems

It is clear from the literature that a number of problems exist in ISD. The first and main problem relates to **the sheer number of different approaches, methodologies and techniques in ISD**. Iivari, Hirschheim and Klein (2001a:180) have identified four ISD paradigms and eleven approaches to ISD, and further point out the "… unabated proliferation of new methods and tools for developing information systems...". They refer to research by Jayaratna (1994) showing the existence of over 1000 ISD methodologies already in 1994. This situation caused Avison and Fitzgerald (2006:566) to call this a "methodology jungle" and to ask: "Where to now for development methodologies?" (Avison and Fitzgerald, 2003:79). Oei et al. (1992:2) call it the "YAMA Syndrome" (yet another modelling approach).

Iivari et al. (2001a) created a hierarchical four-tier framework in an attempt to categorise all of these concepts. The four tiers are the following: ISD paradigms, ISD approaches, ISD methods and methodologies, and ISD techniques.

- **ISD paradigms** are the highest tier and involve issues related to the philosophical underpinnings of ISD where ontology, epistemology, methodology and ethics are considered. Examples of ISD paradigms are functionalism, social relativism, neo-humanism and radical structuralism.
- **ISD approaches** follow from the previous tier and consider goals, guiding principles, fundamental concepts and principles of the ISD process. Examples of ISD approaches are object orientation (OO) and SSM.
- **ISD methods and methodologies** are based on ISD approaches and consider issues such as the relationships between techniques and detailed ISD processes. Examples are ETHICS, OOAD and IE.
- **ISD techniques** consider detailed concepts and notations. Examples include the techniques in UML, CATWOE, ERD and BPMN. ISD modelling is on this level.

Similarly, Adams and Avison (2003) reviewed more than 80 ISD techniques in their paper. Avison and Fitzgerald (2006) highlight the main ISD techniques in use today. They identified 37 techniques grouped into seven categories. For researchers, and especially practitioners, the situation becomes difficult. Which paradigm, approach, methodology and/or technique should be used?

The second problem relates to **the usefulness of methodologies and techniques**. Iivari et al. (2001a) show a considerable questioning of the value of ISD methodologies in the literature. They refer to studies that show that a significant number of organisations do not use standard methodologies and those who do, use in-house-developed methodologies. Similar sentiments are raised by Adams and Avison (2003), showing that methodologies are too complex, need special skills, are not flexible, are expensive and do not necessarily result in better productivity.

ISD techniques, in contrast, are seen more positively and can give any of the following benefits in the development of systems (Adams and Avison, 2003):

- Making the solution to a problem more manageable
- Guiding the problem situation, giving structure and order
- Providing focus and direction
- Providing tools to represent the situation
- Providing a means of communication between stakeholders

In spite of the benefits, Adams and Avison (2003) show that ISD techniques may result in similar confusions as is the case with methodologies and tools. For instance, the same technique can be promoted in several domains, the same technique can have different names in different methodologies, and the same technique can have different modelling symbols in different methodologies.

A third problem that flows directly from the first is the issue of **standardisation and integration**. Because of the proliferation of techniques, methodologies and approaches, no clear universal standards exist. This has caused, among other things, an attempt to unify modelling techniques and languages.

The most successful attempt has been the unification of techniques for modelling software by means of UML. It is very successful in the software modelling area, but has limited capabilities when doing business modelling. Noran (2003) points out the desirability of modelling the software system and its corresponding business, and then shows that UML can only effectively model business if it is complemented by design patterns and specific extensions to capture business processes (Noran, 2004).

In the last few years, a major problem, similar to the proliferation issue discussed above, has arisen in modelling the organisation or business. Many enterprise architectures, languages and tools (like GERAM, IDEF, ARIS, IEM and the Zachman Framework) have been developed, but the language and modelling interoperability between them is very low, meaning that it is not easy to translate information from one enterprise model (or tool) to another. During the International Conference on Enterprise Integration Modelling Technology (ICEIMT) of 1997, this problem motivated the idea of developing UEML.

Fundamentally, UEML should act as an intermediate language that will integrate a wide variety of existing modelling languages. Projects to develop UEML have been sponsored by the European Union's 5th Framework Programme (FP5) on Research and Development, and these projects have been ongoing since the conference (Anaya, et al., 2010).

One of the projects related to UEML is the European thematic network project, Interoperability Development for Enterprise Applications and Software (IDEAS). One of its deliverables was a gap analysis, which highlighted areas for research, technology development and standardisation. The need for more research into enterprise modelling ontology and theory was highlighted in this report. UEML development is currently continued with the Interop-NoE Network of Excellence funded by the EU's FP6 producing versions 2.0 and 2.1 of UEML (Anaya et al., 2010).

This study attempts to contribute, among other things, to shedding more light on the fundamental ontological constructs that must be modelled during ISD.

1.3.2 Business and ISD modelling problems

Gailly and Poels (2007:407) define the goal of business modelling as follows: "... to create semantically faithful and pragmatically usable representations of business domain artefacts (e.g. transactions, processes, value chains)." In this section, business modelling problems are discussed from the viewpoint of general business domain modelling and the more specific viewpoints of business process modelling, requirements modelling and business rules modelling. Finally, problems related to linking business and systems modelling are discussed.

1.3.2.1 General business modelling

Chen-Burger, Robertson and Stader (2000) show that, in spite of a number of significant successes, business domain modelling has a number of problems: availability of expertise, lack of a comprehensive evaluation method (most methods only provide procedures and some measurement criteria), the fact that most methods are informal or semi-informal, the time pressure to validate models by hand, lack of knowledge transfer between developers and managers and very complex dynamic aspects of modelling.

The inability to model various business aspects could have serious results. Gordijn and Akkermans (2002) state that some of the main reasons for the failure of e-business initiatives are that the business ideas were not stated very well (using some form of modelling) and stakeholders could not assess the ideas properly.

Some of the major industry players are working on improving their modelling tools. For instance, IBM introduced its UML profile, a component of Rational Unified Process (RUP), to extend RUP's capability to model businesses (Johnston, 2004).

1.3.2.2 Business process modelling

Russell et al. (2006) indicate that activity diagrams in UML 2.0 have limitations in modelling business processes. They share these limitations with most business process modelling techniques. Specifically, they cannot capture many of the natural constructs found in business processes.

According to Aguilar-Saven (2004), although business process modelling is researched extensively, it still has problems. It is not well structured or classified and there is a lot of confusion related to terminology. Recker et al. (2004) show that although process modelling has improved over the years concerning ontological completeness (using BPMN as the current standard in process modelling) it still has a few potential shortcomings.

Mayr, Kop and Esberger (2007) state that business process modelling and requirements modelling should be based on the same notions and principles because they both occur early in the system development life cycle (SDLC).

1.3.2.3 Requirements modelling

According to Sinha et al. (2009:327), natural language remains the main way of specifying requirements. The adoption of more formal requirements modelling has been slow, mainly because of the "high entry barrier for customer participation".

Ghanbary and Day (2009) state that business requirements, specifically for web development projects, are modelled mostly using development techniques like UML and RUP. They consider current modelling tools and techniques to be incapable of fully capturing business requirements and communicating them to all stakeholders. Some of the specific problems with the current modelling tools and techniques are the following:

- They are unable to present the overall requirements of the system. It is not easy to trace and determine the impact of any change on the whole system.
- The literature of these tools does not adequately explain how to translate pictorial diagrams into actual programming code.
- There is no formal way to ensure that the requirements captured and the requirements that the users want are the same. A major factor is that the current models are not easily understandable by business.
- There is a lack of support for non-functional requirements.
- The current techniques do not have mechanisms to help prioritise processes.

- As complexity of processes increase, the models become more and more confusing to involved parties and many tools have problems in properly showing models over many pages.

Shaker (2010) states that information system (IS) requirements are often defined by users, developers and customers using the features of the system, as well as the functional and non-functional distinguishable characteristics relevant to stakeholders. In current practice, this feature-oriented stakeholder's view is not obvious in development artefacts. He therefore proposes feature-oriented requirements modelling instead of the current goal- and object-oriented modelling.

Winter, Hayes and Colvin (2010:41) attempt "to bridge the gap between an informal language description and a formal model". They do this by incrementally building behaviour trees from functional requirements and merging these trees to form a more complete model.

According to Mayr and Kop (2002:1–2), the main reason why IS projects have incomplete or inadequate requirement models is because "... conceptual models are too complex and abstract as to be easily understood and validated by average users... Requirements engineering therefore, should start with collecting this knowledge and representing it in a way the end user understands and is able to validate."

1.3.2.4 Linking business and system modelling

Tyndale-Biscoe et al. (2002) presented their findings on a project related to improved business modelling of components using UML. Their work attempts to overcome two main problems experienced in extant modelling techniques: optimised linking between the business model and the systems model, and models that are as comprehensible to business experts as they are to system modellers.

Odeh and Kamm (2003) show how difficult it is to find a bridge between a model of a business and the corresponding model of an IT system. The challenge is to find a translation or conversion between the two models. They propose that these two models should be independent, and postulate that a weakness of UML and RUP as

bridging models is that they consider business and IT system models to be dependent. They then illustrate that a direct translation between the two sets of models is difficult, if not impossible, without an intensive interpretation process.

1.3.2.5 Conclusion

In this subsection, business and ISD modelling problems were considered from a number of perspectives. The problems can be summarised into two main issues: providing business stakeholders with techniques and tools for modelling their business environments, issues and requirements that are easy enough for them to use, while providing ISD stakeholders with adequate information to do further systems analysis and design.

1.3.3 Comparison with other disciplines

This study initially started out as an investigation into how ISD modelling can be improved to aid in better communication between all the parties involved in ISD. However, after intensive study of ISD modelling, the researcher realised that the real current problem is not **how** to model IS, but **what** should be modelled and what the underlying modelling theory is. Once one knows what to model, the fundamental modelling theory – the how-to model – would follow logically from that.

When other disciplines are considered in terms of modelling or representation, certain things are clear. Many disciplines do not have the same problems in representation that ISD have. There are various reasons for that, but a fundamental reason is that these disciplines have a clear understanding of **what** they have to represent and how these things are related. As a result of this, they have a consistent, standard way of representing these things.

When considering music, for example, it is clear that, in spite of the numerous notations available in various cultures, they all represent the fundamental dimensions or parameters of music, namely pitch, tempo, melody, harmony, rhythm, loudness, timbre, etc. These fundamental dimensions of music, in turn, are based on the

fundamental properties of sound waves such as amplitude and frequency, the properties of time such as slow, fast and off-beat, and emotional qualities in humans such as joyful, sad and solemn. Because of the standardisation of the Western clef notation, it is possible to play music today composed hundreds of years ago, teach students music theory as well as practice in a consistent manner globally, and to easily translate music notation into actual music, electronically, and vice versa.

Similarly, in architecture, the fundamental, three-dimensional, spatial aspects of a building are represented in the architectural drawings. The architect-designed plan for a building normally consists of a single “blueprint” with complementary plans tightly integrated with the main plan. The building plan is a representation of a three-dimensional artefact that has mainly a space aspect. Therefore, the representation can be done by just considering length, width and height. Interestingly, the two-dimensional “shorthand” version of the plan is the main plan, considering only length and width. The height dimension is taken care of by handling each level as a separate plan (two-dimensional) and by a supplementary plan (two-dimensional) of front, side and back views.

All parties involved with the building of a house work from the same set of plans. For instance, the architect designs the house in the eventual format but without a lot of the detail. The architectural draughtsman adds the detail, like the thickness of the walls and the direction in which the door will actually open. The client approves these plans. A fairly detailed estimate of the building costs can be made based on the plans. The builder uses the plans to determine his quote for the job. All the different subcontractors work according to the plan. The municipality approves the design using the plan. The bank decides on the loan amount, based on the plan.

If one considers representation or modelling for the purposes of ISD, the picture is quite different. The number of techniques, but more so the number of categories, illustrates the problem. Information systems (IS) are seen, and therefore represented, either as data, processes, objects, linguistic actions, and many more concepts. All of these views assume different “building blocks” for IS. Are they all correct or is there a fundamental underlying structure to business and information systems?

1.4 Research approach

The research approach followed in this study is discussed in detail in Chapter 6.

1.4.1 Problem statement and research questions

The problem statement of this study is:

The current modelling techniques do not bridge the gap between business and ISD.

The main research question is:

Can an integrative modelling technique be developed to bridge the gap between business and ISD?

The subresearch questions are the following:

- *Is there a gap between business and ISD that current modelling cannot fill?*
- *What are the fundamental constructs of any integrative modelling technique between business and ISD?*
- *What are the properties and attributes of these fundamental constructs?*
- *What are the relationships between these fundamental constructs?*
- *Can it be demonstrated that the proposed technique does indeed integrate business and modelling techniques better than existing business modelling techniques?*

The research questions are answered in the thesis as a whole, but are also answered explicitly in the following parts of the thesis:

Question	Section/s where answered
Is there a gap between business and ISD that current modelling cannot fill?	1.3.2
What are the fundamental constructs of any integrative modelling technique between business and ISD?	7.2 and 7.3
What are the properties and attributes of these fundamental constructs?	7.2 and 7.3
What are the relationships between these fundamental constructs?	7.2 and 7.3
Can it be demonstrated that the proposed technique does indeed integrate business and modelling techniques better than existing business modelling techniques?	8.3, 8.4 and Appendix A

Table 1-1: Research question map

1.4.2 Research methodology

The research approach followed in this study is design science research. This approach provides another view that complements the positivist and interpretive perspectives of IS research. It distinguishes between natural science and the science of the artificial, and concentrates on phenomena that are created (designed artefacts), rather than objects occurring naturally. Designed artefacts can be, among other things, algorithms, human-computer interaction (HCI) constructs, ISD methodologies and ISD techniques. Design science research involves the following steps (Geerts, 2011):

- Problem identification and motivation
- Defining the objectives of a solution
- Design and development
- Demonstration
- Evaluation
- Communication

Vaishnavi and Kuechler (2004) developed a taxonomy of design science research output (see Table 1-2).

	Output	Description
1	Constructs	The conceptual vocabulary of a domain
2	Models	A set of propositions or statements expressing relationships between constructs
3	Methods	A set of steps used to perform a task – how-to knowledge
4	Instantiations	The operationalisation of constructs, models and methods
5	Better theories	Artefact construction as analogous to experimental natural science

Vaishnavi and Kuechler (2004)

Table 1-2: The outputs of design science research

The artefact of this study is an integrative modelling technique between business and ISD. In terms of the design science research outputs as defined in Table 1-2, this modelling technique can be seen to incorporate the first three outputs: constructs, a model and (to some extent) a method.

Venable (2006) emphasises theories and theorising in design science research and shows, from the literature, the concept of a "kernel theory" that are drawn from natural science, social science and mathematics to provide a theoretical base for the research.

The research in this study comprised two major steps. The first step determined the fundamental underlying concepts and their relationships for business and ISD modelling. This was done using a grounded approach to analysing the main existing business and ISD modelling techniques as well as using linguistics as a kernel theory. This analysis eventually developed into the proposed integrative modelling technique proposed in this research.

The second step was to evaluate and refine the proposed modelling technique using design science research. To evaluate the artefact, it was compared with a number of existing integrative modelling techniques and evaluated based on specific design criteria. Furthermore, various case studies were used to illustrate the application of the proposed modelling technique. The case studies incorporated organisational, business and IT aspects and also the different phases of a system development life cycle.

1.4.3 The basis for the proposed integrative technique

The proposed integrative modelling technique that was used and evaluated in this study is the result of an in-depth qualitative data analysis of existing business and ISD techniques. The main techniques of the past, the main techniques that are currently popular as well as lesser-known techniques were chosen to give as broad a range of techniques as possible. Only representational techniques were considered[1]. The ISD approaches and techniques studied were adapted from Avison and Fitzgerald (2006).

1. *Holistic techniques*: The soft system methodology (SSM) represents the holistic approach, emphasising the techniques such as rich pictures, root definitions (CATWOE) and conceptual models.
2. *Data techniques*: Entity modelling and Structured Query Language (SQL) represent the data approaches to ISD modelling.
3. *Process techniques*: The techniques to represent the process are data flow diagramming, decision trees, decision tables, various IDEF techniques and BPMN. Certain techniques that fall under the process approach, such as action diagrams and entity life cycles, will rather be studied under their object-oriented counterparts.
4. *Object-oriented techniques*: All the diagrams of UML, such as class diagrams, use case diagrams, interaction diagrams, sequence diagrams, state chart diagrams and activity diagrams, are considered as representative of the object-oriented approach.
5. *Project management techniques*: PERT and Gantt charts represent the project management approach.
6. *Organisational techniques*: No representational techniques.
7. *People techniques*: No representational techniques.
8. *Enterprise architecture techniques*: The Zachman framework will represent enterprise architecture techniques.

[1] Two types of techniques are described by Avison and Fitzgerald (2006). Representational techniques describe ways of modelling some universe of discourse, while process techniques describe a set of actions to achieve a certain goal, e.g. the joint requirements planning (JRP) technique describes how to get requirements but not how to represent or model them.

9. *Linguistic techniques*: Language action perspective (LAP) will represent the linguistic techniques.

1.4.4 Scope and limitations

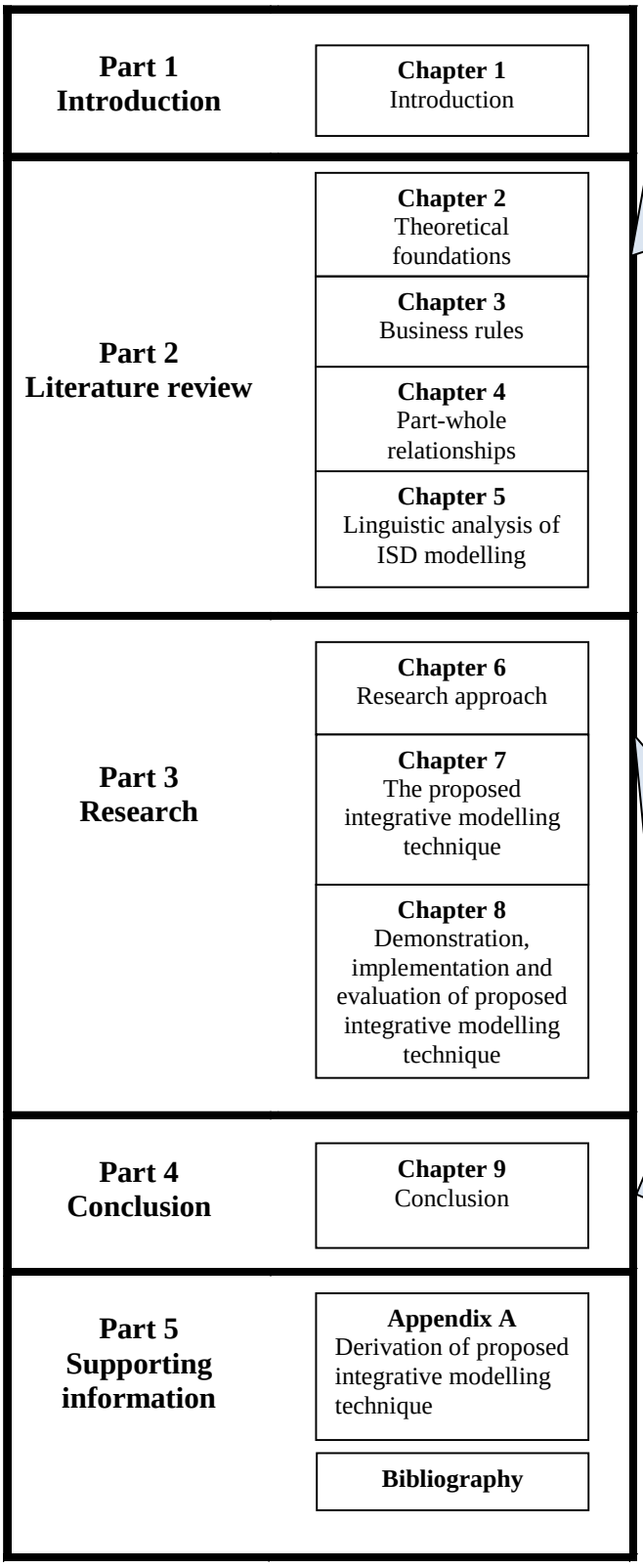
In this study, anything that has a bearing on the various aspects of systems and their development were considered; not only pure ISD issues. It therefore includes the following:

1. Aspects typically found in the SDLC phases of systems
2. The business aspects related to IS like business process management
3. Organisational aspects found in the enterprise architecture domain

Not all the ISD modelling techniques in existence were considered. In the first place space limitations did not allow the inclusion of all techniques in existence. Secondly, because certain techniques are very similar, only one representative technique per category was studied.

The resultant integrative technique was not evaluated in a full-scale, real-life situation, but was evaluated using a comprehensive case study, as well representative examples.

1.5 Layout of thesis



The literature review forms the basis for determining the integrative modelling language proposed in this thesis as well as forming the basis of some "kernel theories" (Venable, 2006). The following main concepts are covered: theoretical foundations (e.g. systems theory), business rules, theory of part-whole relationships, and linguistic analysis of ISD modelling.

Firstly, the two research approaches used in the thesis, namely *grounded theory* and *design science research*, are discussed. It involves looking at the research strategy, research philosophy, research methodology, research design and how it was applied.

Secondly, the proposed integrative modelling language is described.

Thirdly, following the *design science research* process, the proposed integrative modelling technique is demonstrated, a possible implementation is discussed and the technique is evaluated.

The study is concluded and the results evaluated within the bigger picture of ISD. Other areas of research potentially flowing from this study are also discussed.

1.6 Conclusion

In this chapter, it was shown that a number of problems exist in ISD modelling. The first is the sheer number of approaches, methodologies and techniques in ISD. The second is the usefulness of methodologies and techniques. The third is the lack of standardisation and integration. In comparison with many other disciplines, ISD does not have a clear view of what and how to model ISD-related objects and issues.

The purpose of this research is to consider only one of these modelling integration areas, namely the integration between business and ISD modelling, and attempt to develop an integrative technique to bridge that gap. The purpose was not to develop "yet another modelling technique", but to understand the integration problem better and to move towards a better understanding of modelling integration between business and ISD.

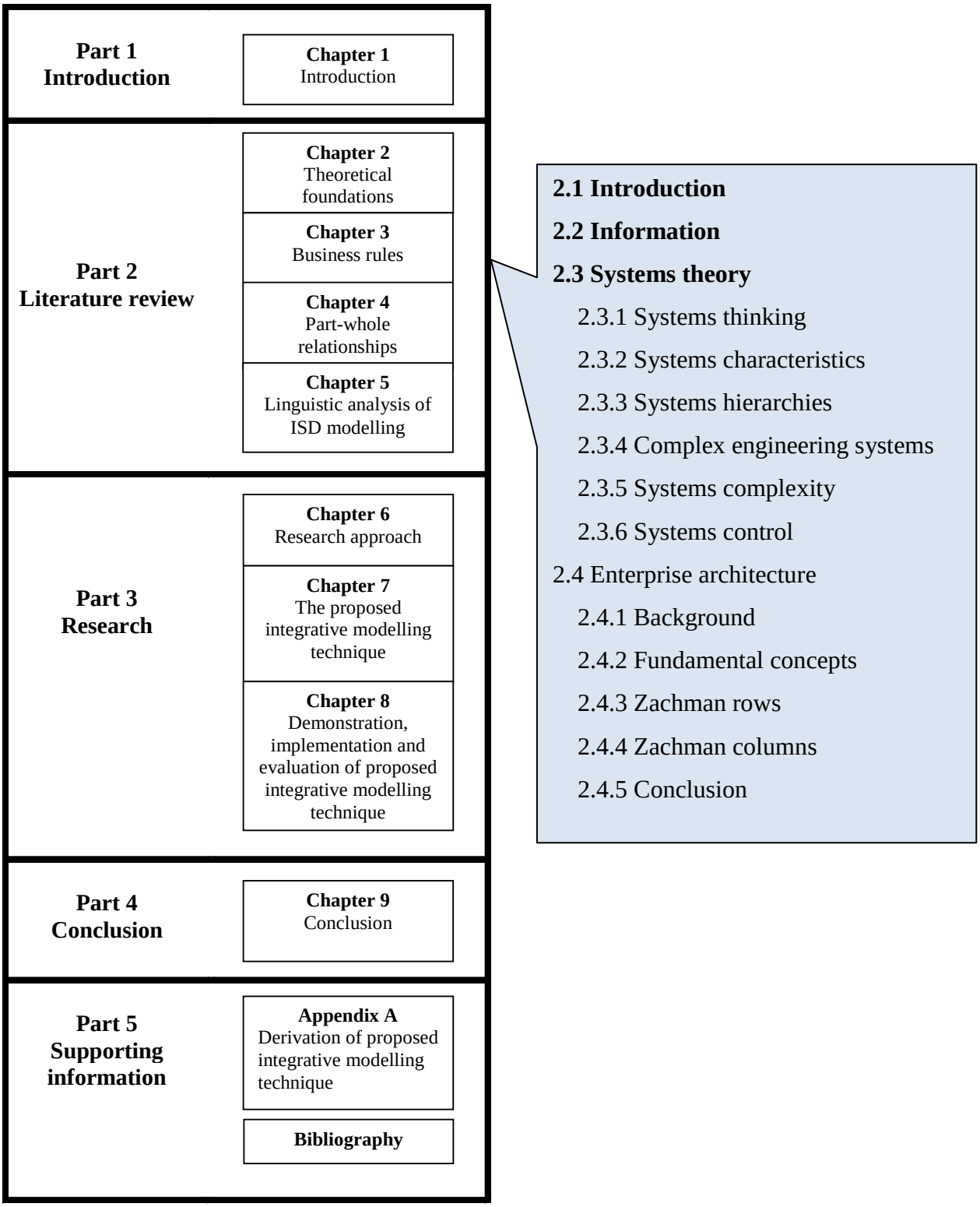
To achieve this goal, a two-pronged research approach was followed. Firstly, to ensure that the technique was based on sound underlying modelling constructs and relationships, a grounded approach was followed in developing the integrative technique, using a representative set of ISD modelling techniques. Secondly, the resultant integrative technique was evaluated and improved using a design science approach.

In order to provide a theoretical foundation for the grounded approach analysis, **Part 2** provides a literature review of a few fundamental underlying areas on the question at hand. These areas are general theoretical foundations (information, systems theory and enterprise architecture), business rules, part-whole relationships, and linguistics.

Part 2

Literature Review

2. Theoretical foundations



2.1 Introduction

In Part 2 of this thesis, a literature review is done as a basis for further research. In this chapter, three theoretical foundations of business and ISD are discussed: information, systems theory and enterprise architecture.

In ISD two different "worlds" come together: business and IS. Both business and IS are fundamentally information "processors", and therefore the basics of information are discussed as the first theoretical foundation.

Business and IS further are related because they can both be considered to be systems. Patching (1990) defines a system, using Collins Standard Reference Dictionary, as "a group of things or parts working together or connected in some way as to form a whole". Significantly, information systems are, by name and definition, systems. Therefore, in this chapter, the second theoretical foundation to be reviewed in the literature is systems theory.

To understand what the components of business and IS are, as well as their relationships, it is further needed to study enterprise architecture as the third theoretical foundation.

In later chapters, the other three foundational areas to be reviewed in the literature – business rules, part-whole relationships and linguistics – are discussed.

2.2 Information

One approach to understanding information is to relate it to semiotics (the study of signs). Signs can be the obvious "visual" signs that most people understand intuitively, like traffic signs, as well as drawings, pictures and photographs. It can also be non-visual signs like words, sounds, objects, acts, odours, gestures and body language, even thoughts. Signs are any objects that stands for some other object, for example, a red robot means "stop" (Chandler, 2007). Signs can be used together with other signs to form coding systems that are used to enable communication between agents.

Some examples of coding systems, with their required number of symbols, are as follows (Gitt, 1997):

- Binary code used in electronic information processing (two symbols)
- Genetic code (the four letters A, C, G, T)
- Decimal code (the 10 digits 0-9)
- Hebrew alphabet (22 letters)
- Latin alphabet (26 letters)
- International flag code (26 flags)
- Chinese writing (> 50 000 symbols)

Coding systems are not created arbitrarily, but they are optimised according to criteria such as pictorial appeal (e.g. hieroglyphics), small number of symbols (e.g. Braille), speed of writing (e.g. shorthand), ease of writing (e.g. cuneiform), ease of sensing (e.g. Braille), ease of transmission (e.g. Morse code) and technological legibility (e.g. bar codes) (Gitt, 1997).

The choice of code also depends on the medium of transmission (Gitt, 1997):

- Acoustic transmission (e.g. natural spoken languages, mating calls of animals, musical instruments)
- Optical transmission (e.g. written languages, technical drawings, flashing signals produced by living organisms like fireflies, flag signals, bar codes, sign language for the deaf, body language, bee gyrations)
- Tactile transmission (e.g. Braille)
- Magnetic transmission (e.g. magnetic tape and disk)
- Electronic transmission (e.g. telephone, radio and TV)
- Chemical transmission (e.g. genetic code, hormonal system)
- Olfactory transmission (e.g. pheromones emitted by some animals)
- Electrochemical transmission (e.g. nervous system).

Information can be transmitted or stored in material media only when a language is available. There are different kinds of languages (Gitt, 1997):

- Natural languages
- Artificial communication languages, such as Esperanto, flag codes and traffic signs
- Formal artificial languages, such as mathematical calculi, chemical symbols, musical notation, algorithmic languages and programming languages
- Special languages found in living organisms, such as genetic languages, bee gyrations, pheromonal languages and hormonal languages

2.3 Systems theory

This chapter consists of the following parts: firstly, the fundamental concepts of systems thinking are discussed and the characteristics of systems are explained. This is followed by a section on different kinds of system hierarchies and one on complex engineering systems. After that, system complexity is considered, followed by the related subject of system control or cybernetics.

2.3.1 Systems thinking

In this section, it is shown how systems thinking went through an evolution moving from closed-system thinking (considering mainly systems in physics and astronomy), to open-system thinking (considering systems in all disciplines) to soft-system thinking (considering human activity systems), and currently to living system thinking (considering different levels of living things). Note that open- and closed-systems thinking together are considered to be “hard” systems thinking.

Introna (1996:33) shows that metaphors are actively used in most ISD methodologies. The major ISD metaphor is that of “system”. Although he acknowledges the benefits of this metaphor, Introna also warns that the “system” metaphor is limited and has probably been overused to the extent that “the map is now the territory”. He wonders what would happen if non-engineering metaphors like “novel” and “battle” would be used to describe IS. However, in this thesis, the major metaphor of “system” will still be used.

2.3.1.1 Hard systems

Although systems thinking has been around for a long time, it was formalised when Von Bertalanffy (1950) observed how different areas of study (such as physics, biology, medicine, psychology and social science) were all changing from studying elementary units or atoms to studying "wholeness". This applied to inanimate things, living beings and social phenomena. These observations gave rise to what is known as the general systems theory (GST). GST identified general system laws applicable to all systems across different fields of study. According to Wang (2004:394), "GST covers the discussion of both mechanistic/close/non-living systems and organic/open/living systems".

For Von Bertalanffy, the father of modern systems thinking, the concept of "wholeness" or "*gestalt*" or "organism" is central in GST (Von Bertalanffy, 1950; Wang, 2004). A number of related concepts flow from this. First is the concept of a dynamic open (vs. a static closed) system, where materials enter and leave a system. Second is the concept of equifinality, where a final state may be reached from different initial states and in different ways. Thirdly, GST has a central assumption called the nonsummativity assumption, which states that the whole is greater than the sum of its parts (Wang, 2004).

GST is applied to almost every discipline of study, for instance, to total quality management (Wang, 2004), organisations (Wang, 2004), criminal justice (Bernard, Poaline III and Pare, 2005), human sciences (Mansour, 2002) and even grounded theory (Stillman, 2006).

In modelling organisations, for instance, GST makes a number of assumptions. Firstly, an organisation is seen as having a goal (externally given desired steady state such as maximising shareholders wealth) and purposes (internally given on two levels: organisational level such as growth and individual level such as increasing personal income). Secondly, organisations are seen as higher-order living (concrete) systems, maintaining order by lowering their entropy with energy and material received through the system boundary. Thirdly, humans (seen by many theorists to be

the basic unit of an organisation) have similar goals and purposes to that of the organisation (Wang, 2004).

According to Woodburn (1988), systems thinking in the Western world developed in the 1960s with the emergence of a number of system-based methodologies described as either systems engineering or systems analysis. These approaches used computer models to explore the behaviour of specific systems with the result that the fields of application for systems thinking were limited to well-structured situations. In these situations, the choice of system and system boundary caused few problems and enough knowledge existed to incorporate theories into the models to explain relevant phenomena.

Woodburn (1988) says that the case for systems thinking can be argued as follows:

- The world around us is complex, but there is evidence of much natural order. There is also man-made order, e.g., rules of the road.
- Man is involved in a quest to enhance the quality of life by controlling events in the world outside and is also driven to seek ways to introduce increasing amounts of order into his environment.
- Until recently, additional new order created through intervention has been at lower levels of organisation or complexity.
- As problems at lower levels are overcome, attention is turned to higher levels of organisation or complexity.
- The idea of a system is an intellectual construct that is considered to generate (intellectual) order.
- Models of these systems can be developed and compared to perceived reality.
- Models can be developed in the form of sets of interrelated activities using ordinary language or in the form of mathematical symbols.

Checkland (1999) provides the four fundamental ideas of systems thinking. These ideas are based on those of Patching (1990), where he clarifies how systems differ from a simplified collection of parts without a common identity and how interaction

between the parts is achieved and controlled. These fundamental ideas are the following:

1. *Emergent properties*. To distinguish an entity from its environment, it must have properties that emerge out of the single whole that are not properties of the collection of parts (Checkland, 1999). Emergence is the display of new attributes in the end product when the component parts of the system are connected (Patching, 1990).
2. *Layered structured*. Entities with emergent properties can have smaller entities with emergent properties which are also systems (Checkland, 1999). Each of these systems forms part of a hierarchy of systems, with the subsystems in turn displaying emergent properties. In man-made systems, this hierarchy is fairly obvious, e.g. a personal computer. However, the extent of the hierarchy will depend on the perspective being taken and any system might be part of a wider system that has some controlling influence, e.g. the personal computer could be part of the communication system of an organisation, or the cultural system of the country (Patching, 1990).
3. *Processes of communication*. The entity must be able to find out about its environment (Checkland, 1999). To function as a whole, there must also be some form of communication between the system components. Each subsystem receives inputs, which stimulate further activity to produce outputs, passing this either to other subsystems or to the environment (Patching, 1990).
4. *Control*. The entity must be able to respond to its environment (Checkland, 1999). Many of the communication messages are concerned with control. Control is the means by which a whole entity retains its identity and/or performance under changing circumstances. Control is normally dependent on feedback about how the system is performing (Patching, 1990).

2.3.1.2 Soft systems

During the 1970s, mainly as a result of work done by Checkland (2000), a different use of the idea “system”, the so-called “soft system”, emerged. At the centre of this approach is the concept of a human activity system modelled by using ordinary language rather than mathematical symbols. It can be used in poorly structured

situations where the choice of system and the delineation of system boundaries are controversial.

The terms **hard** and **soft** are comparative ones used to distinguish between methods of examination that address clearly defined problems and others that are used when the problem is not clear at the outset, and a preliminary investigation is required to identify and select the problems to be solved.

To distinguish between these two approaches, Checkland (2000) referred to them as the **hard** systems approach and the **soft** systems approach. The differences between these two approaches are summarised in Table 2-1 below (using Woodburn (1988) and Patching (1990)):

Hard approach	**Soft approach**
The world is made up of systems.	The world is problematic.
We can study the world systematically.	We can study the world systemically, i.e. by thinking about it by means of a system of inquiry.
An objective for the system can be taken as given.	An objective for the system cannot be taken as given.
Can be applied in well-structured situations.	Can be applied in poorly structured situations.
Systems are seen as physical entities.	Systems are seen as purely intellectual constructs, nothing more.
Asks the question: How do we achieve the objective?	Asks the question: What do we do to achieve an improvement?

Woodburn (1988) and Patching (1990)

Table 2-1: The differences between the hard and soft system approaches

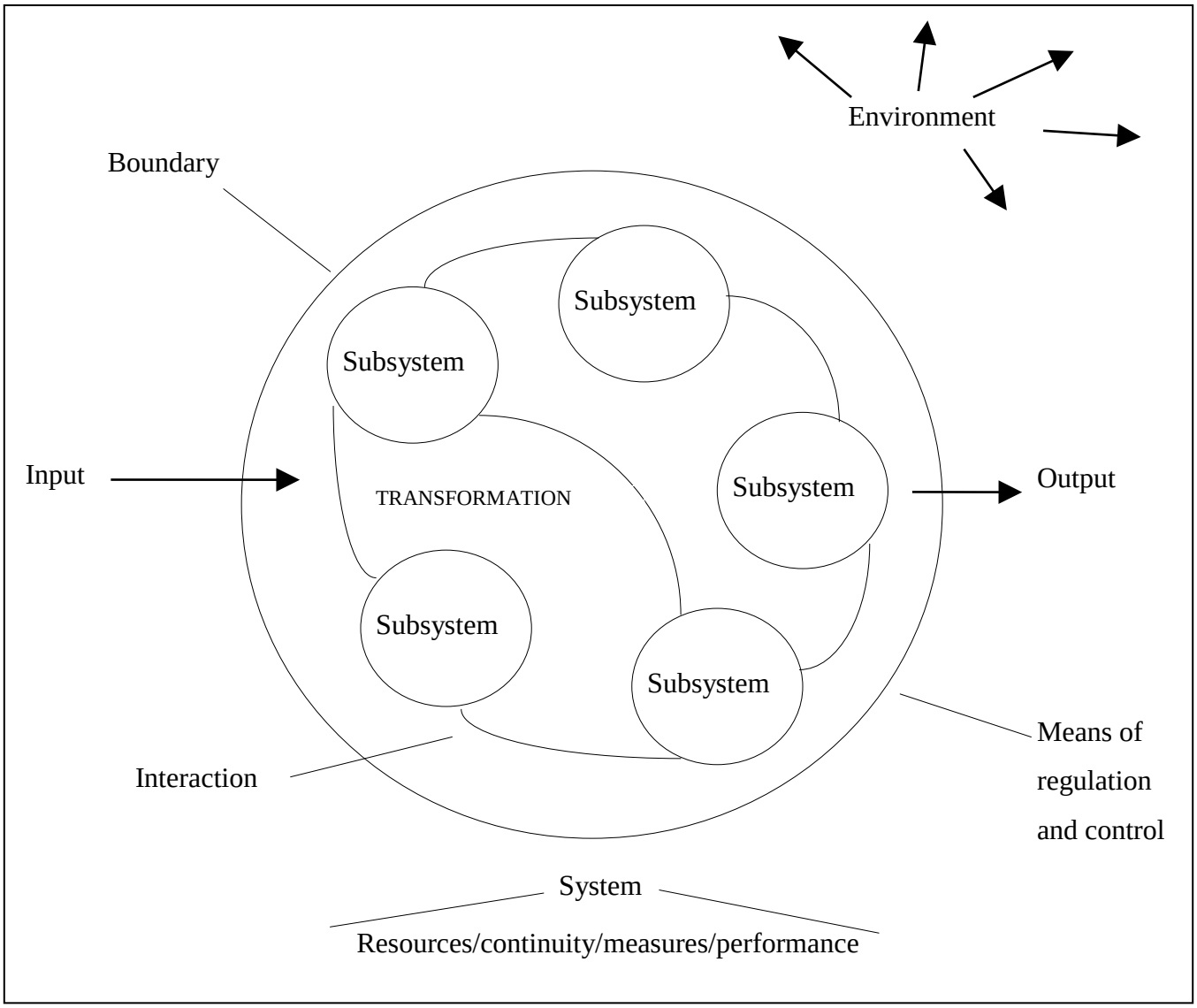
2.3.1.4 Living systems

A more recent theory, the living system theory, in concept already proposed by Miller in 1965, is derived from open systems theory. It proposes hypotheses and processes applicable to eight different levels of living things: "... cells, organs, organism, group, organisation, communities, societies and supranational systems." It is very influential in the study of social systems and has led to many social theories and frameworks (Wang, 2004:394–395).

2.3.2 System characteristics

For the study of ISD, and especially modelling between two systems (IS and organisation), it is necessary to understand the basic characteristics of systems. Patching (1990) discusses these characteristics by using Checkland's formal systems model (see Figure 2-1):

- The system represented by the model has an **ongoing purpose,** i.e. it exists for a reason, and achieves some transformation or change.
- There are **measures of performance** that can be used as a basis for measuring efficiency**,** so that the system can be shown to be effective.
- There is some **mechanism for control** or regulation, and a decision-making process.
- It has components that are themselves systems, i.e. systems can be broken down into **subsystems**. According to Wang (2004), subsystems can be classified into three categories: matter energy, information and combinations of both.
- It has components that **interact**.
- It exists as part of a wider system or systems in an **environment** with which it interacts.
- It has a **boundary** that encloses the area that the regulating mechanism has under control.
- It has **resources** for its own use under the control of the regulating mechanism.
- It has some expectation of **continuity** and can be expected to recover from disturbances.



Patching (1990)

Figure 2-1: The formal systems model – Checkland

2.3.3 System hierarchies

When dealing with IS and organisations, various categories of systems are involved. For instance, human stakeholders working for various organisations are involved in creating IS, based on a specific methodology and set of modelling techniques. All of these underlined concepts are different categories of systems and are categorised in this section.

This section is divided into three parts: The first lists some high-level categories or types of systems. The next two come from Martinelli (2001), who analysed and classified 19 of the most well-known systems taxonomies and hierarchies. He classified these taxonomies and hierarchies according to their utility to management in two main categories: general hierarchies and managerial hierarchies.

2.3.3.1 High-level categories of systems

Patching (1990) distinguishes between four types of systems:

- *Natural systems*: Those that occur naturally in the universe, e.g. galaxies, humans and animals.
- *Designed systems*: These are man-made concrete systems, e.g. computers, central heating systems, jet engines, etc. This category also includes abstract systems such as mathematics, art, music, philosophy, etc.
- *Social and cultural systems*: These systems are formed by human beings coming together, either naturally in families, communities, nations or deliberately in clubs, companies, etc.
- *Human activity systems*: Systems where human beings are undertaking activities that achieve some purpose. These systems would normally include other types, such as social, man-made and natural systems.

Human activity systems can be regarded as open systems, as there is a continual interaction with, and a reliance on, the surrounding environment. Unlike man-made constructions, these can be viewed from a number of different perspectives, each of which would result in a different model. While the subsystems of man-made systems are readily identifiable, these are generally in the form of activities when considering human activity systems. Some of these activities may be observable, but others are taking place as mental processes. Similar problems arise when considering other factors such as communication, boundaries, decision and control mechanisms, interactions with the environment and measures of performance. A human activity system is usually modelled as a series of activities plus an accompanying social system that will have a strong bearing on whether or not certain changes will be accepted (Patching, 1990).

Another way in which systems can be classified is as *concrete systems* in physical space and time, consisting of interrelated objects, *conceptual systems*, consisting of basic units of words or symbols, and *abstract systems*, which have relationships instead of objects as the underlying unit. Concrete systems can be further divided into living (organic) and non-living (mechanistic) systems (Wang, 2004).

2.3.3.2 General hierarchies

According to Martinelli (2001) general hierarchies can further be divided into two categories (see Table 2-2 below): those stressing system complexity and those emphasising system-environment interactions.

Hierarchies stressing system complexity These hierarchies attempt to classify all systems from lowest to the highest level of complexity.	Combining elements from all of these hierarchies gives the following list: • Static • Simple dynamic • Homeostat • Cells • Organs • Plants • Animals • Humans • Groups • Organisations • Communities • Societies • Supranational systems • Transcendental systems
Hierarchies emphasising system-environment interactions These hierarchies consider to what extent the system interacts with its environment.	One set of hierarchies gives, in summary, various degrees of the following: • Open systems • Relatively closed systems • Closed systems Another hierarchy considers the ability of the system to cope with environmental changes: • Autarchic (primitive) systems • Symbiont (bureaucratic, centralised) systems • Dominant (competitive, decentralised) systems • Heuristic (emergent) systems Another hierarchy considers the ruling programme

	of the system and the response to the milieu • Automatic, sequential • Controlled, regulated • Adaptive, self-optimising • Self-learning and self-organising

Martinelli (2001)

Table 2-2: General system hierarchies

2.3.3.3 Managerial hierarchies

According to Martinelli (2001), managerial hierarchies can be further divided into two categories (see Table 2-3 below): those focusing on decision levels and those considering intrasystem and system-milieu interactions.

Hierarchies focusing on decision levels	The first set of hierarchies can be illustrated by the well-known hierarchy in Martinelli (2001): • Production • Operational • Tactical • Strategic A second hierarchy considers the self-government of systems: • Externally governed systems • Systems with embedded goals and controls • Self-learning systems • Self-governing systems • Systems with multiple deciders A third hierarchy considers self-organising behaviour: • Rigidly controlled • Deterministic • Purposive • Heuristic • Purposeful A fourth hierarchy considers the “seven essential elements of any organisation”: • Physical means • Processes and flows • Functions

	• People and leadership • Organisational structure • Policy, strategy, programmes • Identity, mission, distant goals
Hierarchies considering intrasystem and system-milieu interactions Hierarchies are organised according to the increasing complexity of the internal interactions between the subsystems of an enterprise.	The following is a representative example: • Non-systems • Static • Simple dynamic • Feedback dynamic • Multilevel • Autopoietic • Adaptive • Evolutionary

Martinelli (2001)

Table 2-3: Managerial system hierarchies

2.3.4 Complex engineering systems

Information systems can also be seen as complex engineering systems – systems that are designed by humans and have both significant human and technical complexity (Magee and De Weck, 2004). On a more pragmatic level, this view can help us towards a practical understanding of IS. Magee and De Weck (2004) created a taxonomy for the qualitative assessment of complex engineering systems. The attributes of this taxonomy are the following:

- **Degree of complexity** on four levels: part/component, group/subassembly, machine/apparatus, plant/equipment
- **Branch of the economy,** e.g. mining, energy generation and manufacturing
- **Realm of existence**: real or virtual
- **Boundary**: open or closed
- **Origin**: natural or artificial
- **Time dependence**: static or dynamic
- **System states**: continuous, discrete or hybrid
- **Human control**: autonomous, human in the loop or mixed

- **Human wants,** such as shelter, food, transportation, communication, security, longevity and health, entertainment, aesthetic pleasure, education, and social, emotional, spiritual and curiosity
- **Ownership** on three dimensions: single or multiple, for-profit or not-for-profit, and private or governmental control
- **Functional type** (described in the next paragraph)

Functional type is the combination of outputs (operands) and processes (manipulators or actions). Actions can be executed on the following operands:

- **Matter (M)** – physical objects, including organisms that exist unconditionally
- **Energy (E)** – stored work that can be used to power a process in the future
- **Information (I)** – any informational object
- **Value (monetary) (V)** – monetary and intrinsic-value objects used for exchange

The five basic actions are:

- **Transform** or **process** – transform objects into new objects
- **Transport** or **distribute** – change the location of objects
- **Store** or **house** – provide buffers in the network by holding objects over time
- **Exchange** or **trade** – exchange objects mainly via the value operand
- **Control** or **regulate** – drive objects from some actual state to a desired state

The above classification of operands and actions has formed an integral part in the integrative modelling technique created in this study. Understanding that only one of the five types of actions can be executed on any object helped to structure the modelling technique more precisely. For instance, the only actions that can be done on a report (an Information (I) operand) is to transform it (e.g. summarise it), transport it (e.g. courier it to recipients), store it (e.g. place in archive), exchange it (e.g. sell it to another organisation) and control it (e.g. do quality control on the contents).

2.3.5 System complexity

In the previous section, IS was classified as a complex engineering system, and most authors consider ISD to be complex indeed (Xia and Lee, 2005). The many IS project failures also attest to this fact (as highlighted by the Chaos reports of the Standish Group over the years). Similarly, organisations are becoming more and more complex in the rapid changing global economy. In this section, complexity is considered in more detail.

2.3.5.1 Complexity in organisations

Backlund (2002:13) states that complexity is a qualitative concept, although various attempts have been made to define it quantitatively. He defines it as "… the perceived effort that is required to understand and cope with the system." It implies that the complexity of an object can change without the object changing, e.g. when a person gets more experience of the object. He defines the structure of an organisation as complex when one or more of the following is true: (1) it consists of many components, (2) there are many relations between the components, (3) these relations are not symmetric, and (4) the arrangement of the components is not symmetric. He further defines the processes of an organisation to be complex when one or mostly more of the following are true: (1) many "parts" of the organisation are involved in the process, (2) there are many steps in the process and the matter/energy or the information reaches many states or is transformed many times, and (3) there are many different kinds of matter/energy or information involved in the process (Backlund, 2002).

At every point where information is received and not just moved further in an organisation or information system, information can be added, disappear, processed, elaborated, summarised, selected, etc. The efficiency of the information transfer process is indirectly proportional to the complexity of the information system. The less information left from the original reaching the decider, the more complex the information system. Normally information is abstracted as it moves hierarchically upwards in the organisation and made more detailed as it moves downward. One measure of efficiency is the relationship between the total amount of information

received by the highest level of the organisation and the total amount of information created or received by the organisation. Another measure of efficiency is the number of times a piece of information passes through a component that uses it proportional to the average number of components. In summary, the more complex a system, the more connections, the more information, and the more time are required (Backlund, 2002:39).

Lewis (1994) applies chaos theory and its offspring, complexity theory, to organisations and specifically the management of change. A system (e.g. organisation) can be in one of three zones. In the *stability zone* (1), there is stability and predictability, but no change. In the *complexity zone* (2), systems adapt, learn and grow. In the *chaos zone* (3), there is chaos and unpredictability and there are too many changes for learning to take place.

2.3.5.2 Complexity in IS and ISD

Xia and Lee (2005) developed a conceptual framework for ISD project complexity on two dimensions: the distinction between structural and dynamic complexity and the distinction between organisational and technological complexity. Structural complexity has to do with variety and the interdependency of project elements, while dynamic complexity has to do with the uncertainty caused by changes in the environment. Organisational complexity relates to the organisational environments around the project, while technological complexity relates to the complexity of the technological environment, including platform, techniques, languages, methodologies and system integration.

Similarly, Benbya and McKelvey (2006:21) uses complexity theory to define IS as complex adaptive systems that can self-organise, self-optimise and are balanced between order and chaos. They developed a generalised adaptation framework applicable to ISD, based on the following seven “first principles” of system adaptation:

- Tensions in the environment stimulate adaptive order creation (e.g. the conflicting realities of the different stakeholders in ISD).

- If internal complexity is higher than external complexity, adaptive order creation occurs (e.g. building information systems that are able to evolve).
- In a changing environment, higher internal change rates are adaptively advantageous (e.g. quickened learning action loops).
- The design of subunits that are nearly autonomous increases complexity and the rate of adaptive response (e.g. modular and object-oriented design).
- Small positive feedback may result in significant order creation (e.g. development spirals).
- Coping advantageously with multiple causes is needed for complexity (e.g. understanding that not only technological changes, but also organisational and institutional changes influence ISD).
- Rhythmic alternation of causal dominance is better than balance for functional adaptive response (e.g. the influence of design experts vs. user stakeholders).

2.3.6 System control

Closely related to system complexity is the concept of system control. The only way to manage complexity is to control the system in relation to its environment. This is considered in this section.

To control a process, the controller needs to have sufficient internal variety to represent it (the law of requisite variety). The variety of an organisation will always be less than the variety of the environment. Therefore, for organisations to be successful, they should at least have the variety needed to respond to the behaviour the environment is currently exhibiting, or will in future exhibit (Backlund, 2002).

Related to systems theory is the field of cybernetics, which is the study of control systems. A differentiation is made between first-order and second-order cybernetics (Geyer, 1995).

First-order cybernetics is an engineering approach and studies control systems and feedback loops with specific application in controlling intelligent machines. It considers specifically negative feedback loops (naturally or constructed) where the

output of a system is compared with a predefined goal, and if there is deviation from the goal, corrective action is taken. The thermostat of an air-conditioner is a typical example. Efforts to apply first-order cybernetics to, for instance, social sciences were met with resistance, claiming it to be too simplistic and mechanistic (Geyer, 1995).

Second-order cybernetics mostly deals with living systems and specifically includes the observer in the system. These systems have a "will of their own", are more difficult to control, rather use positive feedback loops, and are able to reflect on themselves and on their interactions with their environments. The main concepts of second-order cybernetics are self-organisation, self-reference (knowing about itself), self-steering, autocatalysis (having processes which cause the creation of systems of higher levels of complexity) and autopoiesis (self-production) (Geyer, 1995).

IS and organisations can both be seen as falling under the category of second-order cybernetics and are therefore more difficult to control and manage.

2.4 Enterprise architecture

2.4.1 Background

The Zachman framework is used to describe the architecture of an enterprise because it concentrates more on the contents of such an architecture rather than the process of creating one (as is done in, for instance, The Open Group Architecture Framework (TOGAF)). The original framework (Zachman, 1987) has since been extended (Sowa and Zachman, 1992). The Zachman Framework is considered the de facto standard when specifying architectures and describing the artefacts supporting them by enterprise architecture frameworks like TOGAF (The Open Group, 2007).

The framework describes and specifies the **artefacts** that are important and necessary to build successful information systems (Martin, Roberston and Springer, 2005). An artefact can be classified as any element that is part of a functioning ICT system. It can include any elements such as requirements documentation, manuals or even a software module (Schach, 2004).

2.4.2 Fundamental concepts

The Zachman Framework can also be considered as a reference system containing a categorisation of those artefacts (Martin et al., 2005). The Zachman framework is a taxonomy of system specifications and how they fit together (Sowa and Zachman, 1992).

The Zachman Framework is a two-dimensional matrix consisting of six rows and six columns, giving 36 cells that could contain possible representations of artefacts. The columns consist of questions or uncertainties that must be addressed. The six rows of the framework contain a collection of specific functions performed by the main stakeholders who were part of the process to develop ICT systems. Zachman compares the rows to the stakeholders who are involved in the building of a house. The horizontal dimension or rows consist of a **planner**, **owner**, **designer**, **builder** and **subcontractor**. The vertical dimension of the columns is also known as "focuses". Martin et al. call the questions "interrogatives". The horizontal dimension is also sometimes known as "perspectives" (Zachman, 1987; Martin et al., 2005).

The rest of the cells in the Zachman Framework contain mechanisms that put into perspective all the different role-players (**perspectives**) and the most important facets or characteristics (**focuses**) that must be addressed during the SDLC (see Table 2-4).

Focus

Perspective		Data What?	Function How?	Network Where?	People Who?	Time When?	Motivation Why?
	Planner – Scope						
	Owner – Enterprise						
	Designer – System						
	Builder – Technology						
	Subcontractor – Components						
	Functioning System						

Zachman (1999)

Table 2-4: Zachman rows and columns

The concept of **primitives** has also been identified by Sowa and Zachman (1992). A primitive can be described as the smallest building block of a cell and can be used on its own. Once defined, the primitives can be combined into other more meaningful structures or diagrams (Sowa and Zachman, 1992; Frankel et al., 2003). The concept of primitives is important and will be used to classify examples of artefacts in each Zachman cell.

One Zachman cell could consist of a set of primitives such as narrative descriptions, attributes and types or instances of objects which would serve the purpose of enhancing the description of the cell. Once the primitives have been identified, it should also be possible to store the primitives in a repository for possible future extraction for reporting purposes.

As soon as the primitives of Zachman cells are related together, the resulting structure is defined as a **composite**. This was also identified by Sowa and Zachman (1992) when they described the integration of cells within one Zachman row in order to describe the perspectives of a specific stakeholder. The concept of a composite is described further to show how it is possible to combine cells of different rows together and not only cells within one row, as Sowa and Zachman suggest.

2.4.3 Zachman rows

2.4.3.1 Row 1: Planner or scope

The **scope** or parameters within which the ICT system must operate is decided in Row 1. Concepts discussed here are of a strategic nature and one of the actions is to determine the boundaries of the organisation and how ICT systems will be used within the organisation. The external environment must also be analysed and captured. Any budget constraints must be adhered to. The planner view could also determine how all the components fit together (Sowa and Zachman, 1992; Zachman, 1987).

2.4.3.2 Row 2: Owner or enterprise

All the activities that are important to the business are described in Row 2. The level of obtaining data is high and all the business activities must eventually link to show the business value of what will be achieved if the business activity is performed. Techniques such as business process modelling are important in Row 2. The perspective can show how external policies are interpreted and applied within the organisation (Sowa and Zachman, 1992; Zachman, 1987).

2.4.3.3 Row 3: Designer or system

The level of detail specified in Row 3 remains on a conceptual level and is classified as a **logical level,** since more detail is specified in Row 3 than Row 2. Important to note is that the level of detail in Row 3 is not yet physical. The requirements of the user are specified (Zachman, 1999).

This row is a first step in creating application architecture. System analysis and design techniques will be used effectively in Row 3 (Sowa and Zachman, 1992).

2.4.3.4 Row 4: Builder or technology

The concepts used in Row 4 are inclined to be more of a **physical** nature, together with some logical views. The physical hardware used in the system is specified. The

physical system must be designed together with the connected network, as well as services and devices (Sowa and Zachman, 1992; Zachman, 1987).

2.4.3.5 Row 5: Subcontractor or components

Row 5 would contain the physical concepts that are used to implement executable code. The physical concepts can include any detailed specifications. A **component** is the physical piece of code or software, database or executable that is developed and used by programmers (Schach, 2004). All the commercial off-the-shelf (COTS) products can form part of Row 5.

2.4.3.6 Row 6: Functioning system

The level of detail in Row 6 is also of a physical nature. The actual ICT system has been created and all the concepts created are tangible (Zachman, 1987). It can be argued that Row 6 can be ignored, since it is not part of the architecture of developing an ICT system.

2.4.4 Zachman columns

2.4.4.1 Column 1: Data/What?

Physical and **conceptual things** important to the business are described in this column. These things could be all the nouns used to describe it. An example is “bill of materials” (Zachman, 1987).

2.4.4.2 Column 2: Function/How?

All the **actions** performed by the business are included in this column. The **verbs** used to describe the functions could be indications of all the functions performed by the organisation. It is the process of how important things of the business get transformed by the business (Zachman, 1987).

2.4.4.3 Column 3: Network/Where?

All the **locations** or **places** where activities are performed are described in this column (Zachman, 1987).

2.4.4.4 Column 4: People/Who?

The types of **human resources** that are needed to initiate or perform an activity are described in this column (Zachman, 1987).

2.4.4.5 Column 5: Time/When?

This is an indication of when activities must be initiated, performed and be concluded. **Scheduling** and **sequencing** aspects should be the focus of this column. Specific **time periods** could also be described here. Event modelling could also be used (Sowa and Zachman, 1992; Zachman, 1987).

Column 5 (time) and column 4 (people) have a close correlation with each other, since the parameters within which a task must be completed indicate the number of resources that would be necessary to complete the task. If a 24-hour availability is required, sufficient personnel would be required to address questions and issues that could arise (Sowa and Zachman, 1992).

2.4.4.6 Column 6: Motivation/Why?

All the **reasons** of why activities are important and must be performed are indicated in this column (Zachman, 1987).

2.5 Conclusion

In this chapter, three theoretical foundations were investigated: information, systems theory and enterprise architecture.

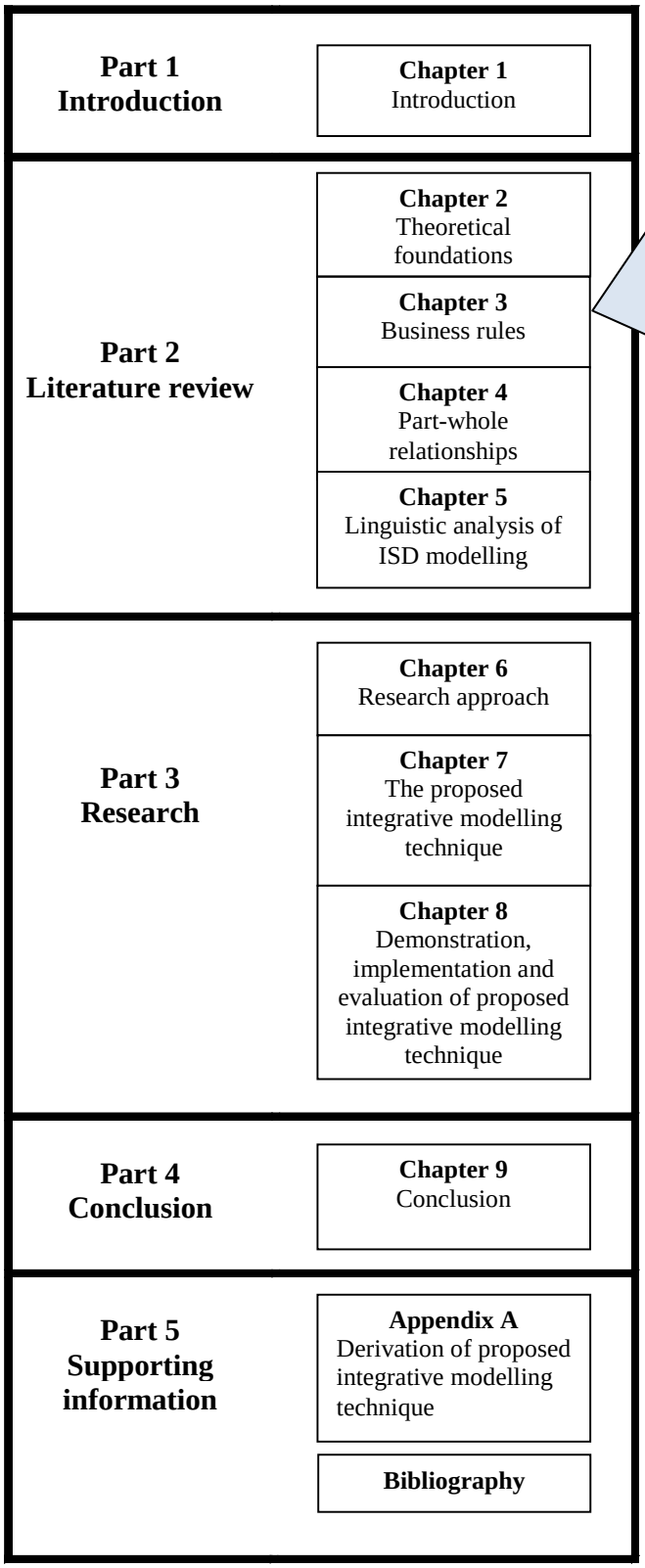
One of the key aspects of business and ISD modelling is information, and an understanding of some fundamental principles of information is needed. Information can be seen as signs forming a coding system with the purpose of communication. Modelling can be seen as a very specific, specialised coding system for communication in organisations and ISD.

System theory is used later in the thesis to explain some of the concepts and constructs of the proposed modelling technique. For instance, both an organisation (called an "institutional actor" in the modelling technique) and an information system (called an "artificial actor" in the modelling technique) can be considered to be systems and exhibit all of the characteristics of systems. Furthermore, the Magee and De Weck (2004) taxonomy provides a very useful way of classifying the actions in business and IS and will be used extensively towards that end.

Enterprise architecture is a formalised way of describing enterprises, their IS, their information and their technology, and therefore formed the third part of the theoretical foundation. The Zachman framework was used as the representative example and illustrated that enterprise architecture can be viewed from two dimensions: the perspectives of various role-players and the main foci of the enterprise.

The main way in which organisations are controlled is by means of **business rules**, providing the systems control discussed in this chapter. In the next chapter, business rules are considered in more detail.

3. Business rules



3.1 Introduction
3.2 Business rules and ISD
3.3 Types of business rules
- 3.3.1 The Business Rules Group's classification
- 3.3.2 Other business rule classifications
- 3.3.3 Summary of business rule types

3.4 Business rule relationships
3.5 Business rule representation
3.6 Conclusion

3.1 Introduction

Business rules are one of the major means by which a system "organisation" controls itself, and a major part of ISD is to elicit business rules and to embed them into an IS subsystem of the organisation with the purpose of enabling the control of a part of the organisation. For instance, business rules embedded in the financial IS, together with the business rules embedded in the manual procedures in the financial department, assures control of the financial subsystem of the organisation.

Business rules have gained prominence over the last few years. They are seen as important assets of organisations that should be managed carefully (Ram and Khatri, 2005). Business rules can also be seen as an important (maybe the most important) link between business and IS (Bajec and Krisper, 2005).

There is no generally accepted standard definition for business rules (Hamza and Fayad, 2005). Many definitions for business rules have been proposed, for instance, that business rules:

- are units of business knowledge (Odell, 1998 in Hamza and Fayad, 2005);
- are constraints or tests designed to maintain the integrity of data (Ross, 1997 in Steinke and Nickolette, 2003);
- are statements that aim to influence or guide behaviour and information in an organisation (Von Halle, 2002);
- define how the business is actually run (Steinke and Nickolette, 2003);
- define or constrain some aspect of a business (Hay and Healy, 2000);
- determine business structure (Hay and Healy, 2000);
- influence the behaviour of an organisation (Hay and Healy, 2000);
- are statements that influence business behaviour towards desired objectives (Steinke and Nickolette, 2003); and
- are assertions that constrain patterns of enterprise behaviour (Morabito et al., 2001 in Bajec and Krisper, 2005).

In essence, business rules are statements that govern the structure and behaviour of various business components.

Some characteristics of business rules are as follows (Bajec and Krisper, 2005):

- They exist in various forms, from simple to very complex and dynamic.
- They can originate:
  - internally, mostly derived from strategic processes, determining the organisation's vision, goals and policies; or
  - externally, from government, industry or specific professional rules as well as natural timeless facts (Herbst, 1996).
- They can be based on explicit (formalised knowledge in the form of principles, procedures, facts, figures, rules and formulas) or tacit knowledge (knowledge that is difficult to see and express).
- They can be found in documents, procedures, policies, regulations, user manuals and IS.
- Explicit business rules are a manifestation of a richer underlying implicit knowledge.

Steinke and Nickolette (2003) consider a business rule good if it has the following characteristics:

- *Declarative*: It is not stated in a procedural manner.
- *Precise*: The meaning of the rule is clear.
- *Atomic*: The rule contains one concept only.
- *Consistent*: There are no conflicting rules.
- *Non-redundant*: No information is repeated.
- *Business-oriented*: It is stated in business terminology.
- *Owned by the business*: Business people are able to maintain the business rules.

3.2 Business rules and ISD

Information systems normally implement a large number of business rules. For example, 627 business rules in a 12 000-line COBOL application and 809 in a 30 000-line COBOL application (Fu et al., 2004).

A typical information system has three elements: an interface (usually GUI), application code/logic and a database. Business rules can be stored in any of these three elements. This can cause various problems with the maintenance of business rules. Steinke and Nickolette (2003) propose that a further layer be introduced to manage business rules.

Business rules also require explicit treatment during ISD to ensure IS agility, otherwise the rules do not reflect real business. This results in applications that do not meet business needs, a lack of documentation on business rules, business rules that are buried in program code, business logic that is hard to maintain and business rules that are hard to control (Bajec and Krisper, 2005). The majority of IS business rules are not explicitly modelled during analysis and design. These rules are only implicitly specified in system models and implicitly embedded in application program code and database structures (Ram and Khatri, 2005).

Updating an implemented set of constraints is not easy, because the mapping between high-level constraints and their implementation in various software artefacts are not explicitly done and maintained (Fu et al., 2004). Therefore, business rules captured in an information system initially can be adequate, but may get outdated of sync later on. There is thus a need for a formal approach towards capturing and managing business rules (Ram and Khatri, 2005).

Various conceptual models have been proposed to capture the meaning and structure of business rules, but most of them only capture a limited range of constraint types. This has given rise to the development of constraint definition languages. These languages are, however, more oriented towards logical than conceptual design and are difficult for users to understand (Ram and Khatri, 2005).

According to Bajec and Krisper (2005), business rule management (BRM) is needed to manage information about business rules' evolution and coordinate their changes centrally.

3.3 Types of business rules

Various types of business rules are identified in the literature and various attempts have been made to classify these rules. The Business Rules Group's classification has become the de facto classification of business rules and forms the basis of the classification used in this study (Hay and Healy, 2000). This classification, together with some other classifications, is also discussed in this section.

3.3.1 The Business Rules Group's classification

The Business Rules Group classifies business rules as follows (Hay and Healy, 2000):

- *Terms*: define a thing or data about it
- *Facts*: indicate connections between terms
- *Constraints (*or *action assertions)*: allow or prohibit actions
- *Derivations* (or *inferences)*: define the transformation of knowledge from one form to another, for instance, formulas

Other similar classifications can easily be related to the Business Rules Group's classification. For instance, Von Halle (2002) identifies at least six different kinds of statements that qualify as business rules:

- Terms that define a noun phrase (term)
- Facts that connect noun phrases into sensible and relevant observations (fact)
- Rules that calculate mathematical results (derivation)
- Rules that constrain the population of facts (constraint)
- Rules that test facts to arrive at a newly discovered fact (derivation)
- Rules that test facts to initiate action (constraint)

Similarly, Perkins (2000) classifies business rules as follows:

- Definitions of business terms (term)
- Data integrity constraints (fact)

- Mathematical and functional derivations (derivation)
- Logical inferences (derivation)
- Processing sequences (constraint)
- Relationships between facts about the business (fact)

3.3.1.1 Terms

The Business Rules Group defines a “term” as a word or phrase that has a specific meaning for the business. A term can be of two types:

- *Business terms*, words or phrases that have specific meaning for a business in designated contexts, e.g. “reservation”, “booking” and “rental request”
- *Common terms*, which are words in everyday language using their commonly accepted meanings, e.g. “car”, “city”, etc. (Hay and Healy, 2000)

Terms can also be classified as follows:

- *Type*, defining abstract categories of things like “car model”, “walk-in rental” and “customer”
- *Literal*, describing instances of things like “General Motors” and “5000”

Business rules are mostly stated in terms of types, but can occasionally refer to specific instances (Hay and Healy, 2000).

Two specific types of terms are the following:

- *Sensors*, which represent the presence of something that constantly detects and reports changing values from the outside world, e.g. temperature reading
- *Clocks*, which are special types of sensors that reports the passage of time; a clock always has one value, the “current time” (Hay and Healy, 2000)

3.3.1.2 Facts

Von Halle (2002) defines "facts" as the relationships between different entities and between an entity and its attributes. Similarly, Steinke and Nickolette (2003) define "facts" as the connections between items.

The Business Rules Group defines facts as associations between two or more terms. For example, the fact "a customer may request a model of car from a rental branch on a date" uses four terms: "customer", "car model", "rental branch", "date". Facts can be stated in many ways, for instance, "each contract may be with a customer" can also be stated as "each customer may be the renter in many contracts" (Hay and Healy, 2000).

The Business Rules Group divides facts in two sets of classifications (Hay and Healy, 2000):

- The first classification distinguishes between a base fact, a fact that is simply stated, and a derived fact, the value of which is computed (mathematical calculation) or inferred from other business rules.
- The second classification defines facts as attributes, where facts are attributes of other terms (e.g. "the colour of the product is blue"), generalisation, where facts are super-types of one or more terms (e.g. "a rental branch manager as a type of employee"), and participation, where facts are the relationship between other terms (e.g. "a rental group is composed of car models").

3.3.1.3 Constraints

One of the most common ways of seeing business rules is to see them as business constraints (Fu et al., 2004; Ram and Khatri, 2005; and Hamza and Fayad, 2005).

Constraints describe the conditions under which an organisation operates. They are normally very volatile as a result of changes to legislative regulations, government policy and business conditions. It is observed that the majority of constraints are defined in terms of business concepts or objects. An example of a constraint is "all

customers whose total business within the last year is greater than x are eligible for y% discount on all orders". This example is stated in terms of business objects such as "customer", "total business", "discount" and "order". (Fu et al., 2004).

Existence rules define how business objects should be created and destroyed (Hamza and Fayad, 2005).

The Business Rules Group defines constraints (action assertions) as statements concerning some dynamic aspect of the business indicating the results that actions can produce: "must", "should", "must not". An action assertion is composed of an anchor object, any kind of business rule and one or more correspondent objects, either another business rule or some specified action. For example: "A car (anchor object – a term) must have a registration number (correspondent object – a fact)" (Hay and Healy, 2000).

The Business Rules Group divides constraints in three sets of classifications (Hay and Healy, 2000):

- The first classification is according to action assertion class. A condition is an assertion that if something is true, another business rule will apply, e.g. "if customer is credit-worthy, give loan". An integrity constraint is an assertion that must always be true, e.g. "a car must be registered". An authorisation defines a specific prerogative or privilege in the form "only x may do y", where x is typically a user and y an action. For example, "only a branch manager of the 'losing' branch may assign a car for transfer to another branch".
- The second classification classifies constraints according to action assertion type. Some of these are enablers and timers: Enablers, if true, permits or leads to the existence of the correspondent object. Timers test, enable or create if a specified threshold has been satisfied, e.g. "if customer is three months in arrears, then …".
- The third classification distinguishes between an action-controlling assertion, which describes what must or must not happen, and an action-influencing assertion, which notifies or serves as guidelines in the human activity system.

Steinke and Nickolette (2003) identify two types of constraints: Integrity constraints, which must always be true, and conditions, which may be true or false.

Ram and Khatri (2005) focus on set-based, static, explicit constraints that restrict the cardinality of a set:

- The attribute constraint specifies the number of attribute values that an entity in an entity class can take. An attribute can be mandatory or optional, and multi-valued or single-valued. For example, “each student must have exactly one student number” implies a single-valued, required attribute, while “each student must have no more than two home addresses” implies an optional, multi-valued attribute.
- The class constraint specifies the number of members in a given class, based on a specific predicate. For example, “there can be between five and seven faculty members with a position of professor in the department”.
- Constraints based on interaction relationships:
  - The interaction participation constraint restricts the number of interaction instances of a relationship that a given combination of entities can participate in. The constraints can be total or partial. For example, “an instructor is allowed to place a maximum of three different reservations for a given book and a course”.
  - The interaction projection constraint specifies the number of distinct entity combinations that can appear in the relationship instances. For example, “the library can accommodate reservations on at most 200 books, regardless of the instructors and the books”.
  - The interaction co-occurrence constraint restricts the number of distinct entity combinations that can co-occur with a given entity combination in an interaction relationship. For example, “instructors are not allowed to reserve the same book for more than two courses that they teach”.
  - The interaction appearance constraint restricts the number of roles that an entity can play and only applies to recursive interaction relationships. For example, “each course has prerequisites and can also be a prerequisite for another course”.

Wan-Kadir and Loucopoulos (2004) define the structure of a mandatory constraint as follows:

> <*subject*> MUST [NOT] <*fact*> *[*IF <*condition*>*]*
> <*subject?* MAY <*fact*> ONLY IF <*fact*>

Herbst (1996) sees the structure of a business rule as the three basic components of event, condition and action (ECA) extended as follows:

- *Event* – when does a business rule have to be processed?
- *Condition* – what has to be checked?
- *Then-action* – what has to be done if the condition is true?
- *Else-action* – what has to be done if the condition is false?

For example:

> *ON (damage-field entered) OR (damage-cause entered)*
> *IF (damage-field = 'private third party insurance') AND*
> *(damage-cause = 'damage of a car in use') AND*
> *(third-party-insurance-type = 'family', 'single' or 'senior')*
> *THEN issue error message "Damages of cars in use are …."*

3.3.1.4 Derivations

Derivations show how and why information is derived from other information (Hamza and Fayad, 2005), apply logic to create new pieces of information (Von Halle, 2002), derive values based on one or more business rules (Steinke and Nickolette, 2003), and infer facts from some other facts (Odell in Ram and Khatri, (2005).

The Business Rules Group defines "derivations" as either (Hay and Healy, 2000) mathematical calculations, e.g. "total cost is charge rate multiplied by hours" or inference, e.g. "the car's rental rate is the same as the car group's rental rate".

Wan-Kadir and Loucopoulos (2004) define the structure of a computation as follows:

<*value*> IS COMPUTED AS <*algorithm*>.

They define the structure of an inference as follows:

IF <*condition*> THEN <*fact*>.

3.3.2 Other business rule classifications

Over and above the types of business rules specified by the Business Rules Group, other types of business rules can also be identified.

According to Perkins (2000), a business statement is a simple declaration in business language stating strategic business rules such as critical success factors, the enterprise mission, goals, policies, objectives, strategies, performance measures, information needs, functions and events.

Steinke and Nickolette (2003) classify business rules on a business statement level as follows:

- *Mandates*: published policies that must be followed, otherwise consequences will be faced, e.g. pay VAT.
- *Policies*: published policies that should be followed to implement the organisational rules, e.g. mission statements. A business policy is a general statement or direction for an organisation (Ram and Khatri, 2005). For example: “We only rent cars in legal, roadworthy condition to our customers.” Each policy may be composed of more detailed policies (Hay and Healy, 2000).
- *Guidelines*: rules followed, depending on some judgment, e.g. management style. Steinke and Nicolette (2003) define guidelines as the “shoulds” of the organisation and mandates as the “musts” of the organisation.

Wan-Kadir and Loucopoulos (2004) define the structure of a guideline as follows:

<*subject*> SHOULD [NOT] <*fact*> *[*IF <*condition*>*]*

Kardasis and Loucopoulos (2004) identify three views for approaching IS analysis: the intentional view, operational view and IS view. These views give rise to the following types of business rules:

- *Intentional rules*: Business rules from a business context perspective. They can be laws, external regulations, principles and good practices specifying the way an organisation does business. These rules are normally expressed as natural language statements.
- *Operational rules*: Business rules from a business process perspective. They prescribe action on the occurrence of some business event or describe valid states of an organisation's information entities. These rules are normally expressed in some rule language.
- *IS architecture rules*: Business rules from an IS implementation perspective.

3.3.3 Summary of business rule types

The following table summarises the different business rule types:

Type	Explanation	Source
Terms	Word or phrases that have specific meanings for businesses in designated contexts.	(Hay and Healy, 2000)
Facts	The relationships between different entities and between an entity and its attributes.	(Von Halle, 2000)
Constraints	Statements about some dynamic aspect of the business indicating the results that actions can produce.	(Hay and Healy, 2000)
Derivations	Business rules derived from other business rules and information.	(Hamza and Fayad, 2005)
Business statements	A declaration in business language stating strategic business rules.	(Perkins, 2000)
Mandates	Published policies that must be followed, otherwise consequences will be faced.	(Steinke and Nickolette, 2003)
Policies	Published policies that must be followed to implement organisational rules.	(Steinke and Nickolette, 2003)
Guidelines	Rules followed, depending on some judgment.	(Steinke and Nickolette, 2003)

Table 3-1: Business rule types

3.4 Business rule relationships

Hamza and Fayad (2005) consider businesses to consist of business objects (like objects and processes) plus business rules. A business change means that either one or more business objects or one or more business rules have changed. Business objects and business rules can be classified according to their stability (probability to change) as stable, partially stable or unstable.

Bajec and Krisper (2005) identify the following business components related to business rules: business goals, processes or activities, ECA structures (meaning, when event happens, and conditions are met, execute activity), business rule descriptions, business terms, business concepts, business actors and resources (e.g. organisation unit, business function or business role). These components are related to business rules in many ways. For instance, business rules support the achievement of business goals, trigger activities, define ECA structures, are described in business rule descriptions, define business concepts, are the responsibilities of business actors and are related to resources. Business rules also relate to other business rules, for instance, one business rule supports another business rule or is in conflict with another.

Rosca and D'Attilio (2001) provide an example of sets of business rules applied to a business action. For instance, business action *calculate discount* can be supported by the following business rules: "*orders > 500 get 30% discount*"; "*orders > 100 receive 15% discount for preferred customers*"; and "*orders < 100 receive 10% discount*". Business actions are seen as fairly stable, while business rules can change frequently.

According to Steinke and Nicolette (2003), a business rule is not a passive, static element. It is triggered by an event, an action, an operation, a condition or a parameter. To really understand a business rule, one should understand the cause and effect on the event.

Poo (1999) defines an event as a set of activities that are performed either fully or not at all (depending on preconditions), after it was invoked by a stimulus (actor or point in time reached by system) and it has an effect on the state of a system by creating or deleting objects and/or changing the state of exiting objects.

Herbst (1996) states that the environment of business rules consists of processes, data model components, organisational units and IS components.

In summary, businesses consist of the following:

- **Events** invoked by some actor or a point in time
- **Actions** (e.g. processes and activities) caused by these events
- **Business rules** constraining events and actions
- **Business objects** (e.g. actors and resources)

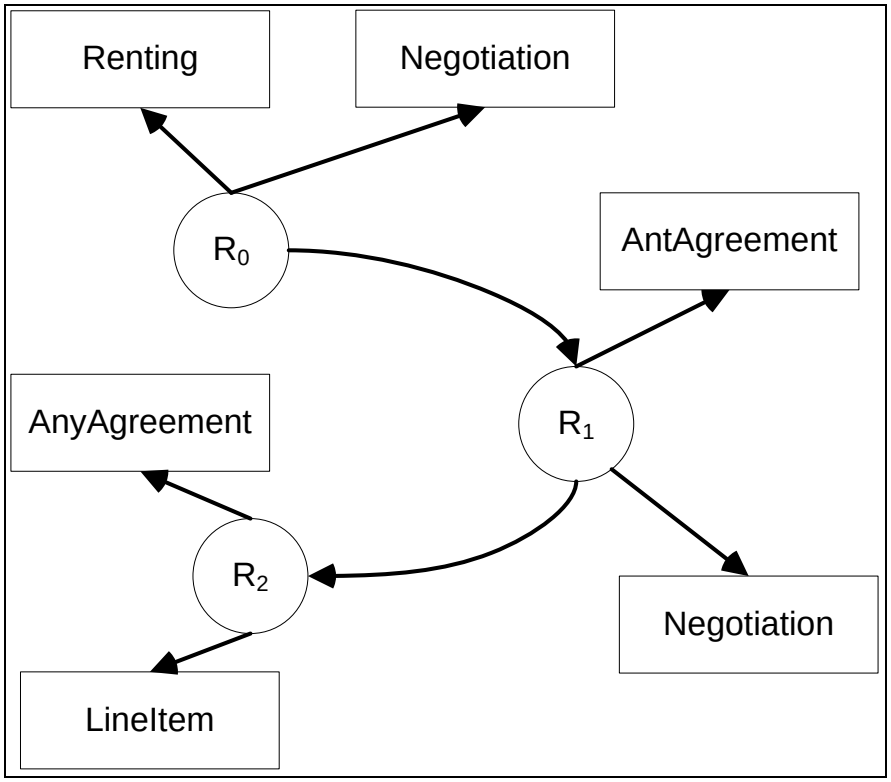
All of these business components are related to each other and to themselves (i.e. business rules are also related to other business rules).

3.5 Business rule representation

Business rules are represented in various formats from natural language statements to formalised rule languages (Ram and Khatri, 2005).

Hamza and Fayad (2005) suggest the reuse of business rules, but also contend that it is complex and hard to achieve. They suggest that to accomplish this, business rules should semantically be abstracted and generalised.

Hamza and Fayad (2005) propose a rule dependency diagram showing the relationships between rules and business objects, as well as between rules and other rules. (See Figure 3-1 for an example of a rule dependency diagram.)



Hamza and Fayad (2005)

Figure 3-1: Example of a rule dependency diagram

Fu et al. (2004) define constraint business rules in terms of structures and constraints and use structure-constraint (S-C) graphs to represent them.

A structure is defined as follows by Fu et al (2004):

- It is an intension for a set of data.
- It can be primitive or a composite of other structures; in other words, not only flat but nested structures are also allowed.
- It has a depth, which is the number of nested structures it consists of.
- It must be acyclical, i.e. not be a component of itself.
- It has a domain:
  - Primitive structure – the set of values from which the structure draws its instances.
  - Composite structure – the Cartesian product of its components' domains.
- It has a state at a specific time – the subset of the domain that the structure has at that time.

For example, the structure that a mobile phone service provider can use to record orders received from its customers is as follows (Fu et al., 2004):

order(customer(id, name, status),
service(network, freeTime),
recommender(id, name, status))

The composite structures are *order, customer, service* and *recommender*; and the primitive structures are *id, name, status* and *freeTime*. The depth of *order* is 2, *customer* is 1 and *id* is 0.

Fu et al. (2004) represent constraints using predicate logic with two restrictions: they only use a small subset of predefined predicates (like *ENUMERATE, EQUAL, ATMOST* and *SUBSUME*) and they use meta-level elements (like *network* and *freeTime*) in predicates. To represent nested structures, they use path expressions (like *service.freeTime*).

For the mobile phone example above, the following are possible constraints (shown in formal predicate logic and informal textual representations):

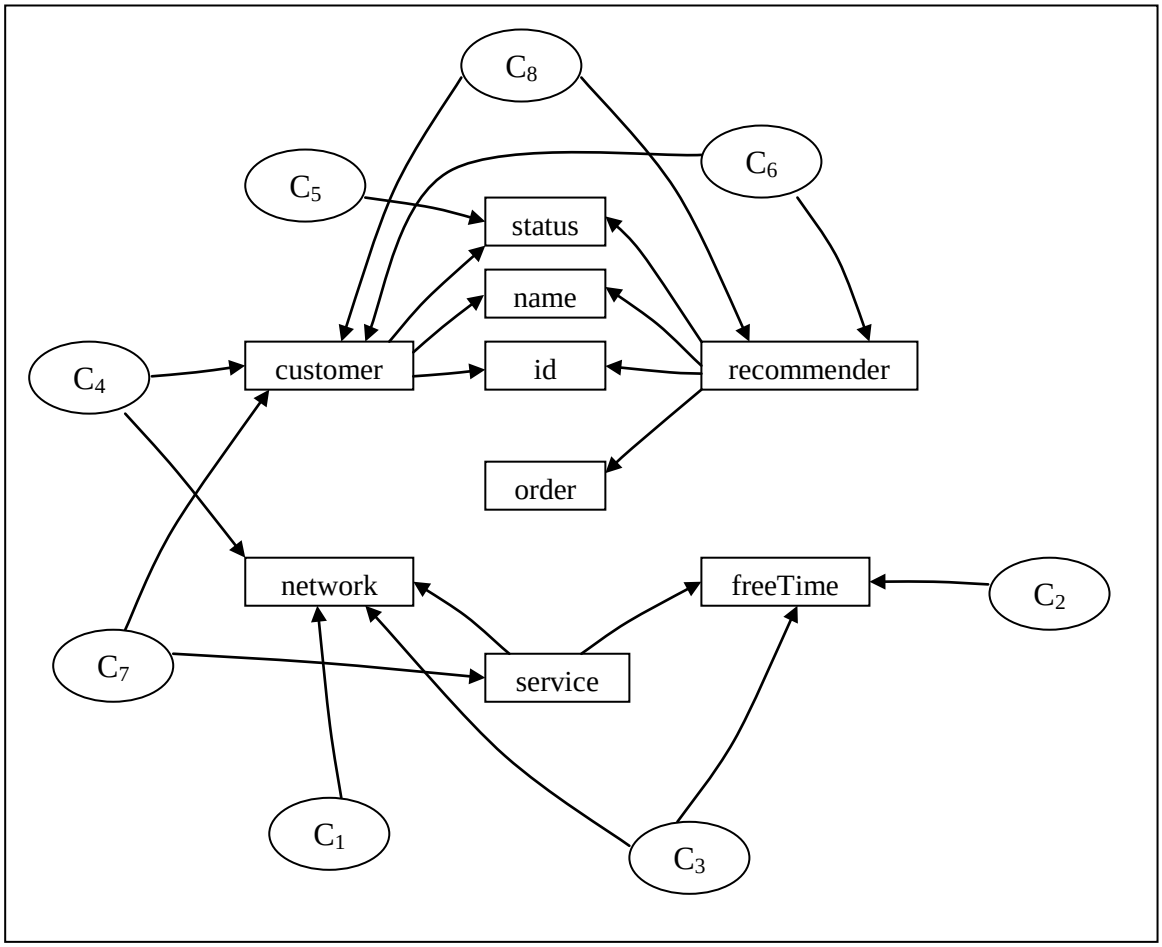
C_1. *ENUMERATE(network, {Vodafone, Orange, O2, T-Mobile})* – The company uses the following networks: Vodafone, Orange, O2, and T-Mobile.
C_2. *ENUMERATE(freeTime,{300, 600})* – The company offers two categories of free talk time: 300 and 600 minutes.
C_3. *EQUAL(service.freeTime, 600)* → *EQUAL(service.network, Vodafone)* – Only Vodafone customers are entitled to 600 minutes' free talk time.
C_4. *EQUAL(order.service.network, O2)* → *ATMOST(order.customer, 10 000)* – The maximum number of O2 services available for ordering is 10 000.
C_5. *ENUMERATE(status, {current, temporary, historic})* – The status of a customer is one of the following: current, temporary or historic.
C_6. *ATMOST(recommender, 3, customer)* – A recommender can recommend at most three services.
C_7. *ATMOST(customer, 3, service)* – A customer can subscribe to up to three services.

C_8. *SUBSUME(customer, recommender)* – A recommender must be an existing customer.

Constraints can be related to structures and these relations can be represented using S-C graphs. An S-C graph is created from a set of structures (S) and a set of constraints (C) (see Figure 3-2).

Constraints can be related to structures as follows (Fu et al., 2004):

- Directly related if the constraint is related to the structure itself, e.g. C_5 is directly related to *status*.
- Indirectly related if the constraint is related to any components of the structure, e.g. C_5 is indirectly related to *customer* and *recommender*.
- Implicitly related if it does not exist, but can be deduced from existing constraints, e.g. from current constraints we can deduce: C_9. *ATMOST(order.recommender, 9, order.service).*



Fu et al. (2004)

Figure 3-2: Example of an S-C graph

3.6 Conclusion

Business rules are statements that govern the structure and behaviour of various business components. They are very important in the context of analysing, designing and developing IS, but also in businesses in order to model strategic, tactical and operational business rules.

Business rules are mostly classified as terms, facts, constraints (action assertions) or derivations. Some research has gone into the structure of the various types of business rules. Business rules are linked to other business objects such as actors and resources, actions such as processes, activities and events invoked by actor and time.

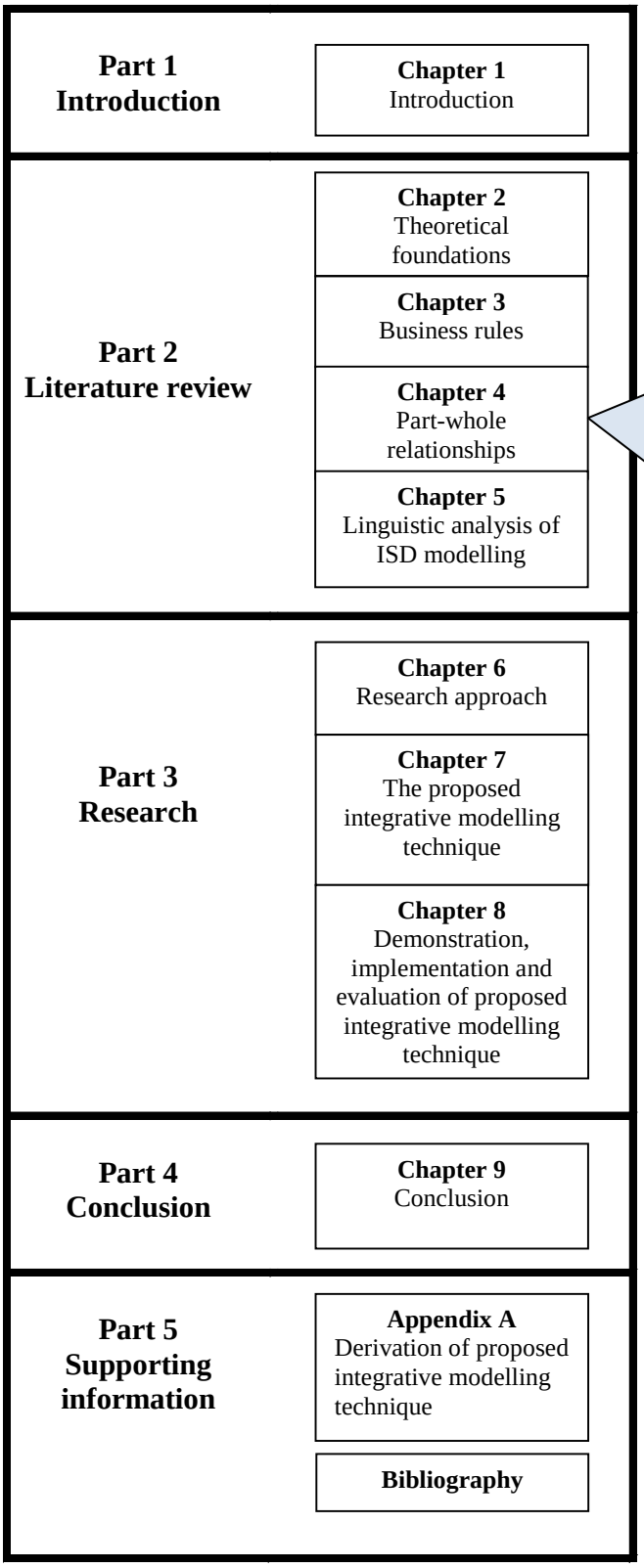
Businesses are defined by business rules and these business rules are incorporated into IS as follows:

- **Business objects** (e.g. actors and resources) are described by *terms*, *facts* and *derivations*, which are mostly embedded in databases and files.
- **Events** and **actions** (e.g. processes and activities) are described by *constraints* and are mostly embedded in programs and manual system procedures.

The goal of this study is to develop an integrative modelling language between business and ISD. Because business rules are such an important link between business and ISD, it must be possible to represent all types of business rules using this integrative technique. A major part of evaluating the proposed integrative technique in this research would be to consider the relative ease with which business rules can be modelled by it and how easy it is to convert these models into ISD models.

Another issue that is important to understand in the relationships between business and ISD is the concept of part-whole relationships where objects consist of subobjects. This is discussed in the next chapter.

4. Part-whole relationships



4.1 Introduction
4.2 Background
4.3 Overview
4.4 Part-whole classifications
- **4.4.1 Classical mereology and classical extended mereology**
- **4.4.2 The Opdahl et al. frameworks**
- **4.4.3 The Gerstl and Pribbenow framework**

4.5 Part-whole relationships in ISD
4.6 Conclusion

4.1 Introduction

The issue of part-whole and related relations is a very important one in IS modelling. Guizzardi (2011:1) states: "Parthood is a relation of fundamental importance in conceptual modelling." Kilov and Sack (2009:101) postulate: "One of the most important concepts in system thinking is that of a composition relationship." Modelling techniques use various mechanisms to indicate part-whole relationships. For instance, in both data flow diagrams and IDEF0 diagrams, one context-level process, representing the whole system, gets expanded into two or more processes, which in turn get expanded until one has primitive processes that cannot be expanded any more. In UML, systems and modules are represented by packages containing all of their constituent parts, and in class diagrams aggregation and composition can be modelled with specific symbols.

In this chapter the Bunge-Weber-Wand ontology will, firstly, be used to provide a context and introduction to the discussion on part-whole relationships. This ontology describes various ontological constructs, among them *relationships*, and then specifically the *composite* construct, which relates directly to part-whole relationships. Secondly, the related concepts of mereology, part-whole relations, partonomies, ontology and taxonomy are discussed to provide an overview of part-whole relationships. Thirdly, three specific part-whole frameworks are discussed to illustrate some older and more recent thinking on part-whole relationships. Finally, part-whole relationships in IS modelling is also discussed.

4.2 Background

Wand, Storey and Weber (1999) developed an ontology based on Bunge's ontology that they use to formally analyse the relationship construct. This ontology (called the Bunge-Weber-Wand ontology) is a key reference work in the theory of object orientation. It can briefly be summarised as follows (describing various ontological constructs):

- The world is made up of **things** (concrete or conceptual) that possess **properties** (properties of conceptual things are called **attributes**).
- Humans conceive things by means of **models** of things (which are conceptual things). This implies that different models of the same thing can exist and that not every property will necessarily be represented by an attribute.
- A thing is an instance of a certain **type** of thing.
- A **functional schema** can be defined as a view of a set of related things. For instance, a person can be viewed as a student, employee or lecturer.
- A set of values of attribute functions of a thing at a certain point in time represent the **state** of the thing at that point in time.
- Wand (1996), in an earlier work, also defined the construct "event" as the transformation of a thing from one state to another. Events can be identified as either external (due to the actions of other things) or internal (due to transformation inside the thing).
- Restrictions on the possible combinations of the components of a functional schema are called **laws.**
- Things **interact** when one may cause changes to the other.
- **Mutual properties** are properties that exist in two or more things. Interaction implies a mutual property in the interacting things. For example, a company that has employees must have a mutual property such as "work-for-company".
- Two things may associate to form another thing. Things are **composite** if they are the combination of two or more things, otherwise they are considered **simple.** For instance, the things in a composite can be *component-of* or *part-of* the composite thing.
- Wand (1996), in an earlier work, also defined the construct "system" as a composite made of interacting things. This implies that the **environment** of the system is the things not in the system with which the system interacts and that a **subsystem** is a part of another composite system.
- "A property of a composite thing is **inherited** if and only if it is a property of any of its components; otherwise, it is **emergent,**" (Wand et al., 1999:504). A related postulate states that every composite has emergent properties.
- Relationships between things are categorised by some researchers as either **topological** (also called connection) relationships, for instance, a husband and

wife interacts, or **mereological** (also called part-of relationships), for instance, husband and wife are part of the family.

See Table 4-1 for a mapping of ontological constructs to conceptual modelling constructs.

Ontological construct	**Commonly used construct**	**Proposed generic conceptual modelling construct**
Thing	Entity Object	Instance
Property	No direct representation	No direct representation
Attribute representing an intrinsic property	Attribute (of an entity or an object)	Attribute (type: intrinsic)
Attribute representing a mutual property	Relationship (binary or n-ary) Reference attribute	Attribute (type: mutual)
Interaction attribute representing a binding mutual property	Relationship Reference attribute Message connection Service request	Attribute (type: mutual, binding)
Class	Entity type Object class	Class
Kind	Entity type Object class	Class
Natural kind	Object type	Class
Simple thing	Entity Object	Instance (type: simple)
Composite thing	Aggregate entity or object	Instance (type: composite)
Connection attribute representing a binding mutual property	Relationship	Attribute (type: mutual, binding, topological)
Component-of attribute representing a binding mutual property	Relationship Part-of	Attribute (type: mutual, binding, mereological)

Wand et al. (1999)

Table 4-1: Mapping ontological constructs to conceptual modelling constructs

All ISD modelling involves (using Bunge-Weber-Wand ontology terminology) firstly identifying ontological constructs like actors, processes, data stores and objects. Secondly, it involves identifying the relationships between these ontological constructs. For instance, *a client* (actor) *can have* (relationship) *many invoices* (objects). This chapter specifically considers the class of relationships where one

entity is part of another entity, the so called part-whole relationships (mereological relationships in the Bunge-Weber-Wand ontology).

The reason for specifically considering these relationships is because of the difficulty of identifying part-whole relationships in practice. One of the more challenging parts of a modelling a system is to determine how to decompose the different parts of a system in a consistent, repeatable, non-subjective way. Many of the traditional decomposition techniques have no clear rules on how to decompose systems consistently. Wand et al. (1999:495) state: "While both entities and relationships are fundamental to conceptual modelling, relationships prove to be more problematical."

In the next section, various fields of study related to part-whole relationships are discussed to provide a more formalised background to understanding how the various modelling constructs are related.

4.3 Overview

Under the heading of part-whole, the following concepts are of interest: mereology, part-whole relations, partonomies, ontology and taxonomy.

According to the Stanford Encyclopaedia of Philosophy (Varzi, 2010:1), "mereology" is "the theory of parthood relations: of the relations of part to whole and the relations of part to part within a whole". It involves any part or portion of a specific entity or object. Some examples are as follows:

- *The screen is part of the laptop* (the part is attached to the whole).
- *The laptop case is part of the laptop* (the part is detached from the whole).
- *The front part of the office is mine* (the part is arbitrarily demarcated within the whole).
- *Mauritius is part of Africa* (the part is disconnected from the whole).
- *The corner is part of the table* (the part is immaterial).
- *The four corner points are part of the circumference of the square* (the parts are immaterial).

- *The first half of the game was the best* (the part is a temporal subset of the whole).
- *Humanity is part of personhood* (the part is a property of the whole).
- *Metal is part of cars* (the part is a material constitution).
- *Eggs are part of pancakes* (the part is a mixture composition).
- *Preparing lectures is part of being a good lecturer* (the part is a conceptual inclusion).

According to Gerstl and Pribbenow (1995), **part-whole relations** are important in various disciplines such as linguistics, knowledge processing, philosophy, psychology and artificial intelligence. Specifically in philosophy, part-whole has been considered a fundamental ontological relation.

Bernauer (1996) explained that the part-whole relation has a long tradition in the medical domain, because medical concepts normally refer to anatomical objects and their parts. These are normally represented by means of standard terminology or classification systems. These systems are many times combined with a coding system (see for example Figure 4-1).

T	Topography axis
T1	Muscoloskeletal system and soft tissues
T11	Bones of shoulder girdles, pelvis and extremities
T114-T116	Bones of upper extremity
T1141	Humerus
T11412	Corpus humeri
T11401	Pars proximals corpus humeri

Bernauer (1996)

Figure 4-1: Concept ladder

Part-whole relations are also important in IS modelling. In OO modelling, data modelling and similar fields, there are many relationships where one object is part of another object, i.e. a whole-part relationship (Opdahl, Henderson-Sellers and Barbier, 2001a).

A **partonomy** is the structure of an object and the parts associated with it. It is represented by a tree with no more than three levels: the whole, its parts and parts of its parts. Partonomies differ from **taxonomies** in that they represent part-whole relations while taxonomies represent *is-a* relations (Gerstl and Pribbenow, 1995).

Ontology is traditionally the philosophical study of the nature of being and existence (Kayed and Colomb, 2005). More recently, the term "ontology" refers to something more specific. Although there is a lot of debate on the definition, one of the most cited definitions is the one by Gruber (1995:908): "An ontology is an explicit specification of a conceptualisation", where a conceptualisation is "an abstract, simplified view of the world that we wish to represent for some purpose". In its most basic form, these ontologies provide a shared vocabulary representing a specific domain's knowledge (Mihoubi, Simonet and Simonet, 1998).

Although "parts" and "whole" seem complementary, they differ in some important aspects. "Part" is a binary, relational concept, while "whole" is a unary, predictive one. Something can be a part only if it is part of a whole, while a whole does not need parts to be a whole. But mostly a whole is made up of parts structured in such a way that the whole acquires *integrity*. Integrity is not well understood, but seems to be dependent on kind, respect and relevance. The part-of relation, on the other hand, is independent of integrity (Eschenbach and Heydrich, 1995).

Next, some part-whole frameworks and classification systems are discussed to provide a more detailed understanding of the concept "part-whole relationship".

4.4 Part-whole classifications

In this section, three part-whole relationship frameworks are discussed. They are classical mereology together with classical extended mereology, the Opdahl et al. framework and the Gerstl and Pribbenow framework.

4.4.1 Classical mereology and classical extended mereology

Classical mereology consists of the following definitions (D1–D6) and axioms (A1–A3) (Eschenbach and Heydrich, 1995):

- [D1] x is *part of* y iff (if and only if) x is discrete from everything y is discrete from.
- [D2] x is a *proper part* of y iff x is part of y and y is not part of x.
- [D3] x and y *overlap* iff they have a common part.
- [D4] x is the *sum* of some entities iff x is discrete from exactly those entities which are discrete from them.
- [D5] x is the *product* of some entities iff x is the sum of all their common parts.
- [D6] x is an *atom* iff it has no proper part.
- [A1] x is discrete from y iff x and y do not overlap.
- [A2] If x is part of y and y is part of x, then x and y are identical.
- [A3] For any entities, their sum exists.

Classical extended mereology (Gerstl and Pribbenow, 1995) is an axiomatic system characterising the *part-of* relation:

- **Existence** – If A is part of B, both A and B exist.
- **Asymmetry** – If A is part of B, B is not part of A.
- **Supplementation** – If A is part of B, B has a part C such that there is no X which is both part of A and part of C (i.e. B has a part C disjoint from A).
- **Transitivity** – If A is part of B and B is part of C, then A is part of C.
- **Extensionality** – Objects with the same parts are identical.
- **Existence of mereological sum** – There exists a unique mereological sum S for any non-empty class of existing individuals.

4.4.2 The Opdahl et al. framework

Part-whole relationships can be analysed by considering the characteristics of relationships themselves or by considering the characteristics of the relationships in

the concrete problem domain that is being modelled (ontological analysis) (Opdahl et al., 2001a).

Opdahl et al. (2001a) developed a framework of characteristics of whole-part relationships (see Table 4-2). They classify the characteristics into four groups:

- *Primary characteristics* are necessary conditions for a relationship to be a whole-part relationship. Primary characteristics must be Boolean (true or not).
- C*onsequential characteristics* are logical consequences of one or more of the primary characteristics.
- S*econdary characteristics* are not necessary for all whole-part relationships; they are rather used to identify and distinguish between different types of whole-part relationships. Secondary characteristics do not have to be Boolean.
- *Dependent characteristics* are only possible in specific combinations with other secondary or dependent characteristics.

Type	**Characteristic**
Primary characteristics	**Whole-part:** The *part-of* relationship has the following three attributes: (a) *Idempotent*, which means that adding the same part more than once in an aggregate does not make a difference. (b) *Commutative*, meaning that the parts of an aggregate are unordered. (c) *Associative*, which means that the order in which parts are added to an aggregate does not make a difference.
	The aggregate object having one or more resultant properties: An aggregate must have at least one property that results from the properties of its parts. For example, mass for physical object, because all parts have mass.
	The aggregate object having one or more emergent properties: An aggregate must have at least one property that does not result from the properties of any of its parts. For example, intelligence in a person, because none of the body parts have intelligence.
	Irreflexiveness at the instance level means an object cannot be its own part.
	Antisymmetry, and therefore asymmetry, at the instance level: *Asymmetry* means that two objects cannot be reciprocally part of one another.
	Antisymmetry at the type level means that two different classes cannot both play the role of aggregate in a whole-part class relationship with another class.
Consequences of primary characteristics	**Propagation of operations to part and ownership of the part:** System-oriented characteristics.

Type	Characteristic
	Abstraction: If the aggregate belongs to another class, then all its parts also do. For example, "head" cannot have another "head" as its part. In contrast, "groups of people" can be part of (larger) "groups of people".
Secondary characteristics	**Lifetime relationship** refers to whether an aggregate object is created and destroyed before, simultaneously with or after its parts. Because the lifetimes of an aggregate and its part must overlap, there are nine possible combinations.
	Transitivity or intransitivity: Transitivity means that if thing A has thing B as part and B has thing C as part, then A must also have C as part.
	Shareable or unshareable parts: A thing can be part of more than one aggregate thing at the same time. For example, "tree" can simultaneously be part of "estate" and "residential area".
	Configurational relationships between parts: Refers to whether there are *structural* or *functional* relationships between parts of an aggregate. Structural relationships represent permanent spatial relationships between the part things. Functional relationships represent that parts combine to produce resultant properties (including laws) of the aggregate.
	Separable or inseparable parts: A thing can exist without being part of a particular aggregate. For example, a keyboard can exist before and after it is used as part of a PC.
	Mandatory or optional parts: An aggregate can exist without having a part of a particular class. For example, a car may or may not have a radio but must have an engine.
	Mutable or immutable parts: Can an aggregate have a particular part replaced by another, equivalent part without losing its identity?
Characteristics dependent on secondary characteristics	**Propagation of the delete operation** refers to whether deleting the aggregate will also delete the parts.
	Separable or inseparable aggregate refers to whether an aggregate can exist without having a particular part (the same as secondary characteristic above).
	Existential dependency refers to the dependency of a thing on a particular instance and not only a class of another thing. This is a characteristic of general relationships and not only whole-part relationships. Applied to whole-part relationships, it refers to (a) whether a whole can exist without a particular part, (b) whether a part can exist without being part of a particular aggregate, or (c) whether the parts of an aggregate object can exist independently of one another.
	Coverage of the parts refers to whether all the physical matter represented by the aggregate is also represented by at least one of its parts.
	Detached or intersecting parts: refer to whether parts are detached or intersecting the whole.

Adapted from Opdahl et al. (2001)

Table 4-2: The revised Henderson-Sellers and Barbier's framework of characteristics of whole-part relationships

4.4.3 The Gerstl and Pribbenow framework

Winston et al. (1987), in Gerstl and Pribbenow (1995), developed a classification system of six meronymic (part-whole) relations. This classification is based on three criteria:

- Functional – parts are restricted, by their function, in their spatial or temporal location, e.g. *handle-cup*.
- Homeomerus parts are of the same type as their wholes, e.g. *slice-pie*.
- Separable – parts can, in principle, be separated from the whole, e.g. *handle-cup*, while inseparable parts cannot, e.g. *steel-bike*. This criterion is only applicable where both the part and the whole are physical objects.

Every criterion above can be applicable or not, giving eight possible combinations of these three criteria. In the Gerstl and Pribbenow (1995) framework, however, only six of these corresponding relations are discussed in more detail:

- **Component/integral-object** – Functional and separable, e.g. *handle-cup, punchline-joke*.
- **Member/collection** – Separable, e.g. *tree-forest, card-deck*.
- **Portion/mass** – Homeomerus and separable, e.g. *slice-pie, grain-salt*.
- **Stuff/object** – E.g. *gin-martini, steel-bike*.
- **Feature/activity** – Functional, e.g. *paying-shopping, dating-adolescence*.
- **Place/area** – Homeomerus, e.g. *Everglades-Florida, oasis-desert*.

Based on the framework of Winston et al., Gerstl and Pribbenow (1995) developed a "common-sense theory of part-whole relations". Their approach not only considers the compositional structure of the whole as in previous theories, but also relations that result from the application of external criteria.

Part-whole relations brought about by the compositional structure of the whole are as follows:

- **Quantity/mass:** The part has no compositional structure, but it can be divided into homogeneous quantities by applying an arbitrary quantitative measure, e.g. *100 grams of rice in the pan, five minutes of the soccer game, the majority of the votes*.
- **Element/collection**: E.g. *three of the dozen apples, one of her holidays*.
- **Component/complex:** The parts are distinguished on the basis of their spatiotemporal arrangement with respect to the whole and/or on the basis of their contribution to some function of the whole, e.g. *engine of the car, head of the department*.

The following are part-whole relations independent of the compositional structure of the whole:

- **Segments:** Parts that are created by applying an external scheme. Normally the external schemes are spatial (if the whole is a spatial object, or can be represented as a spatial entity). For example, *the upper part of the house*. The most useful external scheme is the one-dimensional scheme that basically divides a "line" in *beginning*, *middle* and *end*. The "line" can be a street, a queue of people, a story or a factory process. A three-dimensional cube scheme can be applied to solid physical objects where distinctions must be made between top/bottom, front/back and left/right. For example, *the lower right corners of the fridge, the back panel of the bookcase*.
- **Portions:** Parts that are created by applying one or more property dimensions to the whole, e.g. the dimension colour as in *the red parts of a painting* or the dimension valuation as in *the sad parts of the story*.

4.5 Part-whole relationships in ISD

According to Artale, Franconi, Guarino and Pazzi (1996) the normal way of interpreting the role played by single attributes in a class description as *'has-a'* has potential problems. It makes it difficult to distinguish real part-whole relations like the *door* of a *house* from real attributes like the *colour* or *price* of the *house*. They argue that part-whole relations cannot simply be modelled by ordinary attributes.

Artale et al. (1996) propose that the minimum requirements of a conceptual model to capture the ontological nature of both parts and wholes are as follows:

- Explicit introduction of (complex) wholes in the model. There are two ways to model part-whole relationships: *Implicitly*, where the relationships between objects are modelled using attributes and the knowledge concerning the whole is spread among different objects. *Explicitly*, where the relationships between objects are modelled in new objects and the knowledge of the whole is held in the whole itself. The explicit approach has benefits in terms of reusability, understanding and extendibility.
- Clear distinction between parts and other attributes of a whole.
- Built-in transitivity of parts. Transitivity is the most discussed algebraic property of whole-part relations. Transitivity states that if *x* is part of *y* and *y* is part of *z*, then *x* is part of *z*. This is true in examples like the finger is part of the hand, the hand is part of the body, and therefore the finger is part of the body. But it does not hold in examples like the arm is part of the musician, the musician is part of the orchestra, and therefore the arm is part of the orchestra.
- Possibility to refer to parts by generic names.
- Capability to express "integrity" relationships between parts and the whole. Relationships between parts and whole can be seen from the following perspectives:
  - Vertical relationships
    - Existential dependence relationships. *Rigidly dependent* means that an individual cannot exist without another individual, e.g. "person" and "brain". *Generally dependent* means that an individual cannot exist without another "type" of individual, e.g. "person" and "heart".
    - Property dependence relationships. These include properties the whole inherits from its parts and vice versa. It also includes properties of the parts that are systematically related to the properties of the whole, e.g. the weight of a single part is always less than the weight of the whole.
  - Horizontal relationships
    - Constraint relationships between the parts characterising the integrity of the whole.

According to Shanks, Tansley and Weber (2004), using an object/entity class to represent a composite solves many of the problems experienced when the composite is only represented as an association. For instance, a person may be a step-parent in one or multiple marriages (but not other marriages). This fact cannot be represented easily in models where marriage is not shown as a relationship class.

4.6 Conclusion

In this chapter, the Bunge-Weber-Wand ontology (Wand et al., 1999) was discussed. It describes ontological constructs such as things, properties, attributes, types, states, functional schemas, events, law, interaction, mutual properties, composites, systems, topological relationships and mereological relationships. This provides an ontological context for the part-whole relationship (called a mereological relationship in the ontology).

Secondly, the related concepts of mereology, part-whole relations, partonomies, ontology and taxonomy were discussed to provide an overview of part-whole relationships.

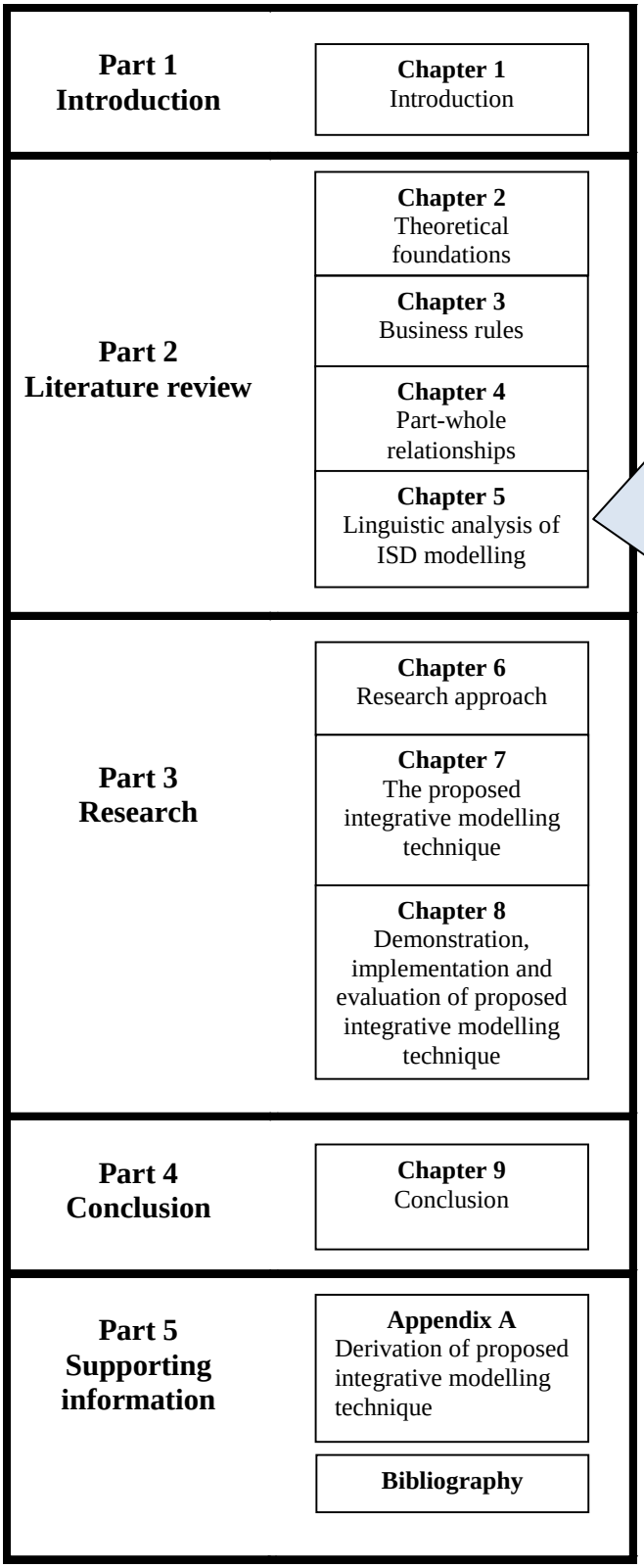
Thirdly, three part-whole frameworks or classifications were discussed, namely the classical mereology as well as classical extended mereology, the Opdahl et al. framework, and the Gerstl and Pribbenow framework.

These frameworks can be applied to IS modelling by considering various types of modelling constructs and the context in which the aggregates are formed (these applications will be developed in more detail in the grounded analysis). For instance, if human agents are considered in the context of IS, they cannot be decomposed into parts. (In a biological or medical context, human actors are composed of parts like respiratory, skeletal and cardiovascular). On the other hand, if institutional actors (like organisations) are considered, it is clear that they can be composed of various parts like employees, branches, buildings and departments. The aggregate object (the organisation), for instance, has resultant properties like size and emergent properties like industry type. Employees can be seen as unshareable, separable, mutable,

mandatory parts of an organisation. The relationship between an organisation and an employee can be classified as complex-component.

Finally, part-whole relationships in ISD were discussed. The major issue emanating from that discussion is that part-whole relationships must be seen as entities in their own right and must not only be seen as associations.

5. A linguistic analysis of IS modelling



5.1 Introduction

Modelling is closely linked to linguistics and language. Most modelling techniques use linguistic constructs to help analysts to identify IS modelling constructs.

In the literature, numerous authors link linguistics and IS modelling (Chen, 1976; Capuchino, Juristo and Van de Riet, 2000; Carter, Long and Truex, 2007; Leppanen, 2006; Charaf, Rosenkranz and Holten, 2010).

In his seminal work on entity relationships, Chen (1976) has shown that there is a correspondence between ERD constructs and natural language. He shows that a common noun corresponds to an entity type, a proper noun to an entity instance, a transitive verb to a relationship type and an adjective to an entity attribute.

Capuchino et al. (2000:26) propose a conceptual modelling method based on the idea "… that there is some relation between the linguistic world, in which the user need is represented, and the OO conceptual world, in which developers represent the above need."

The purpose of this chapter is to expand the statements made by Chen, Capuchino et al. and others, and to do an extended comparison between linguistics concepts and IS modelling.

5.2 Linguistics and IS modelling

In this section, fundamental linguistic concepts are related to IS modelling. Linguistics is divided into a number of different areas. The areas of morphology, syntax, semantics and pragmatics are directly applicable to IS modelling, while other areas like phonology and phonetics are not (the last two areas have to do with sounds of language and the sounds of human speech, respectively (Stabler, 2010), not contributing to the issue of modelling directly).

Please note that a number of standard works on linguistics and natural language processing were used to provide the information in this chapter (Stabler, 2010; Shinghal, 1992; Valeika and Buitkiene, 2003; Kornai, 2007; Haspelmath, 2001). Where required, other works were used and specifically referenced.

5.2.1 Morphology

Morphology is concerned with one of the most fundamental units of linguistic structure, namely the word.

5.2.1.1 Words and morphemes

Words are constructed out of morphemes, i.e. any part of a word that cannot be broken down further into meaningful parts. Compare for instance the words “class” and “classes”. The morpheme “class” cannot be broken down any further, while the word “classes” consists of a base morpheme, “class”, and a plural morpheme, “es”. In a similar manner, we have the words “schedule”, “schedules”, “scheduled” and “scheduling” related to the base concept of “to schedule”.

Linguistic concept	**IS modelling link**
Words and morphemes	Not all words used in a specific area to be modelled will be used for IS modelling. For instance, when verbs are specified in process and object modelling, only the first-person present tense format is normally specified, e.g. “order” and not “orders” or “ordered”.

5.2.1.2 Lexicon

In a natural language, such as English, all words in the language are described in a general dictionary that represents the language’s lexicon, i.e. a list of definitions of every word in the language.

Linguistic concept	IS modelling link
Lexicon	In a business or information system, situation words can have very specific meanings not necessarily as defined in the dictionary. Therefore, most methodologies recommend defining all words (often called terms) that have a context-specific meaning. For instance, in normal use "client" and "customer" can be seen as synonyms, but in a specific organisation "client" could mean "a client who has no account with us", while "customer" could mean "a client who has an account with us".

5.2.1.3 Lexical categories

Words can firstly be grouped as open-class or closed-class words. Open-class words (also called content words) belong to the four major lexical categories of noun, verb, adjective and adverb. Closed-class words (also called grammatical or function words) belong to the minor lexical categories.

(a) Open-class and closed-class words

The set of open-class words tends to be quite large and "open-ended", i.e. new words can be created and added almost unlimitedly. Just consider all the new words created fairly recently as a result of advances in information technology, e.g. "cellphone", "email", "spam", "hacker".

Closed-class words belong to the minor lexical categories of articles ("the", "a"), demonstratives ("this", "that"), quantifiers ("all", "most", "some", "few"), conjunctions ("and", "or"), comparatives ("more", "less"), prepositions ("to", "from", "at", "with") and pronouns ("I", "you", "she", "her", "them"). The set of closed-class words tend to be relatively small and additions or changes to it is unlikely to happen often (e.g. it is highly unlikely that changes or additions to the lexical group of articles will take place in the next few years).

Linguistic concept	IS modelling link
Open-class words vs. closed-class words	There are an unlimited number of concepts as described by entities, objects and processes, but a more limited number of "reserved" words, like "if", "and", "or", and "for every".

(b) Nouns

Nouns denote persons, places or things, e.g. *the man walked to London*. The things can be those we perceive through our senses or those we can conceive in our minds as ideas. Things also include animals. Nouns have certain important properties: (1) number, i.e. singular or plural, (2) case, marking categories such as subject (nominative case), object (accusative case), and ownership, origin or association **(**genitive case**)** and (3) gender in languages like German and French.

Various categories of nouns, called **genus**, can be identified:

1. **Proper nouns:** Names of specific persons, places or things, such as "Shakespeare", "Canada", "Mount Everest", "Susan". These nouns are written beginning with an upper-case letter.
2. **Common nouns:** Names of non-specific persons, places or things, such as "city", "horse", "women", "milk", "ambition", "thought". A common noun cannot be a proper noun and vice versa.
3. **Count nouns:** Those that can be counted, such as one "man", two "men", etc. When used in sentences, these nouns are frequently preceded by words like "a", "an", "each", "every" or "many".
4. **Mass nouns** (or non-count nouns): Those that cannot be counted. These nouns do not usually have a plural form. Examples are "dirt", "foam", "water", "honesty", "homework", "steel". When used in sentences, these nouns are frequently preceded by words like "much", "more", "little" or "less". Some nouns can be used both as count and mass nouns, e.g. "she pulled out two hairs" (count noun), "she cut her hair" (mass noun).

5. **Collective nouns:** Name of a group with the members of the group sharing some characteristics: an “army” (of soldiers), a “crowd” (of people), a “flock” (of geese), a “herd (of cows) and a “team” (of players). A collective noun is usually considered to be singular.
6. **Compound nouns:** Those that were originally written as two or more words. Either a sequence of separate words, a sequence of hyphenated words or one word derived from merging the original sequence of words, for example, “funny bone”, “mother-in-law”, “blackboard”.
7. **Concrete nouns:** Names of tangibles like “book”, “board”, “plane”, “crowd”, “water” and “Mount Everest”.
8. **Abstract nouns:** Names of intangibles like “ambition”, “fragrance”, “honesty”, “integrity”, “truth” and “thought”. An abstract noun cannot be a concrete noun, and vice versa.
9. **Living nouns:** For example, “plant”, “shrub”, “man”, “woman”, “boy”, “girl”, “colt”, “filly”.
10. **Animate nouns:** For example, “man”, “woman”, “boy”, “girl”, “colt”, “filly”.
11. **Human nouns:** For example, “man”, “woman”, “boy”, “girl”.
12. **Masculine nouns:** For example, “man”, “boy”, “colt”.
13. **Feminine nouns:** For example, “woman”, “girl”, “filly”.
14. **Neuter nouns:** For example, “plant”, “shrub”.

Note that genera 1–8 are normally found in grammar books (Stabler, 2010:58–60), while genera 10–14 are used to create natural language processors (Shinghal, 1992:145–146). A noun may be of more than one genus, for example, a colt is a common, count, concrete, living, animate, masculine noun.

Linguistic concept	**IS modelling link**
Noun	Nouns are very important in IS modelling and many modelling techniques recommend using nouns to identify modelling objects, for instance, entities (ERD) and classes (UML).
Number of a noun	An important part of data and object modelling is to identify the one or the many parts in a relationship. For instance, *one* customer can have *many* orders.

Linguistic concept	**IS modelling link**
	There is also the concept in object orientation of the design pattern called "singleton", a class with just one instance, for example, the class *Current president of the country* will always just have one instance.
Case of a noun	The nominative and accusative cases of a noun can be used to clearly make a distinction between subjects and objects. The genitive case can be used to identify whole-part relationships like aggregation in class diagrams, and entity-attribute relationships in ERDs. For instance, the product's components and the client's name.
Genus of a noun	The whole concept of categorisation per se is important. It relates to, for instance, class hierarchies. Proper nouns are rarely used in modelling (unless there is in reality only one of a type, e.g. *The Reserve Bank of South Africa*). Proper nouns will mostly indicate the value of an attribute. A more generic noun indicating the relevant role will rather be used. For example, *Finance department* should be seen as a specific instance of *department*.
Count and mass nouns	The distinction between count and mass nouns has no direct use in IS modelling, but in practice most nouns are count nouns.
Collective nouns	Indicate special relationships like aggregation, e.g. *project team* implies *team members*.
Abstract and concrete nouns	There is no distinction in modelling between the two types. Both types are handled equally.
Human and neuter nouns	Human nouns indicate possible actors and agents, while neuter nouns indicate mostly the objects of actions.

(c) Verbs

Action verbs portray actions, e.g. "he walked slowly forward", while **existence verbs** indicate states of existence, e.g. "Absa is a bank".

Different types of verbs can be identified concerning their transitivity. **Transitive** verbs take one noun phrase after them, like "the client paid his account"; **intransitive** verbs do not take any noun phrases after them, like "John laughed"; while **ditransitive** verbs take two noun phrases, like "the Pope proclaimed Elizabeth the queen".

Another categorisation of verbs concerns auxiliary and main verbs. **Auxiliary verbs** are a closed set and includes forms of the verb "be" ("is", "am", "are", "was"), forms of the verb "have" ("have", "has", "had"), forms of the verb "do" ("do", "does", "did") and modal auxiliaries indicating possibility, necessity and obligation ("can", "could", "will", "would", "shall", "might", "may", "must"). **Main verbs** are verbs like "run", "walk" and "sing".

A verb has six properties. Like a noun and a pronoun, a verb has a **person** (first, second or third) and a number (singular or plural). For example, "walks" is a third person singular verb. In a sentence, the person and number of a verb is the same as the person and number of its subject.

The **tense** of a verb indicates the time of the action or the state of existence portrayed by the verb. There are three tenses: past, present and future. Within each tense there are four aspects: Simple – action just happens. Perfective – action completed in past, present or future. Progressive (or continuous) – action continues in past, present or future. Perfective progressive – a combination of perfective and progressive.

The **voice** of a verb denotes the relationship of the verb with its subject. It can be active or passive. In active voice, the subject does the action portrayed by the verb: "Archie showed the book." In passive voice, the action is done to the subject: "The book was shown by Archie."

The **mood** (or mode) of a verb tells us about the attitude and understanding of the speaker or writer about the action or state of existence portrayed by the verb. A verb can have three modes: indicative, imperative and subjunctive. The indicative mood makes a statement or asks a question. For example, "she will be a singer", or "will she be a singer"? The imperative mood issues a command, an exhortation or a request, e.g. "show your book" or "have mercy on me". The subjunctive mode

expresses (1) certain stock expressions, like “be that as it may”, “come one, come all”; (2) a condition expressed contrary to fact, like “if I were you, I would have greeted her” (in reality, I am not you); (3) a desire, recommendation or a requirement by using words like “ask”, “demand”, “essential”, “important”, “insist”, “move”, “necessary” and “obligatory”. For example, “I insist that he show his book”. The subjunctive is gradually disappearing in practice, except in stock expressions.

Transitive verbs can sometimes occur without an overt direct object, but there is almost always an implied, unexpressed, covert direct object. For instance, “he ate” implies that he ate food and not something else.

There are a few transitive verbs that have little information and depend on the rest of the predicate to provide meaning. For example, “John does my taxes”, “she does her nails”, and “they are having a meeting”.

Linguistic concept	**IS modelling link**
Verb types	Action verbs (plus a noun phrase) are mostly used to describe action-related modelling constructs like functions, programmes, use cases, processes, etc. For example, the “*OrderProduct* screen” or the “*PrintEmployeeDetail* report”. Note that on a higher level, action-related constructs are defined by nouns, for example, “payroll system”. Existence verbs are indicative of relationships between entities, e.g. “the cashier is an employee”. Main verbs are used mostly, while auxiliary verbs are seldom used, except for modal auxiliaries that are used in business rules, e.g. “all orders must/should be authorised by the department manager.”
Person and number of a verb	Because one works with roles, the person and number of a verb is not relevant and most modelling techniques indicate verbs to be first person singular.
Voice of a verb	Only the active voice is used in modelling. Passive voice sentences are basically never used to model and are transformed to active.

Linguistic concept	IS modelling link
Tense and aspect of a verb	This is related to time and state and can be indicated in different ways. Modelling will mostly be done in simple present tense. If this is an as-is or a to-be picture of the system, it will be indicated by the context.
Mood of a verb	Most modelling will be in indicative mood. Imperative mood will be used to specify business rules and instructions, for instance, “only authorised managers can approve leave application”.
Transitivity of the verb	Most verbs will take one object, for example, “update client information” and “order product”. Implied direct objects are normally made explicit. If a transitive verb with little information, such as “does” occurs, it normally indicates a function or process on a higher level in the decomposition hierarchy. For instance, “manager does day-end procedure” is most probably on a higher level than “manager prints day-end report”.

(d) Adjectives

Adjectives specify the attributes of a noun or pronoun, e.g. “the tall girl danced”. When an adjective is part of a noun phrase, it is called an attributive adjective, e.g. “the fat lady”. When an adjective is not part of the noun phrase and it complements a verb, it is called a predicative adjective, e.g. “the lady is fat”.

Linguistic concept	IS modelling link
Adjectives	They relate mostly to the values of attributes of a corresponding entity or object, e.g. “red” is the value of attribute “colour” of entity/object “rental car”.

(e) Adverbs

Adverbs modify verbs ("he sang loudly"), adjectives ("a very tall building"), other adverbs ("unbelievably quickly") and sentences (sadly, he died). Semantically, adverbs indicate when, where, how or to what degree.

Adverbs can be of the following types:

1. **Adverbs of manner** modify a verb to tell how an action is done, e.g. "he waited eagerly".
2. **Adverbs of place modify** a verb to tell where an action is done, e.g. "she lives near the sea".
3. **Adverbs of time** modify a verb to tell when or how long an action is done, e.g. "he cried yesterday" and "he cried unendingly".
4. **Adverbs of frequency** modify a verb to tell how frequently an action is done, e.g. "he cried once".
5. **Adverbs of degree** modify a verb to tell how much an action is done, e.g. "he nearly had an accident".
6. A **sentence adverb** modifies a sentence to tell about the writer's comments, e.g. "frankly, he is a snob".
7. **Adverbs of focus and viewpoint** explain the focus or viewpoint of a sentence, e.g. "he doesn't like pudding, especially Christmas pudding" and "financially, things are going well".
8. **Truth adverbs** express what the speaker knows about the truth of statement, e.g. "maybe she is lost" and "the athlete allegedly took steroids".
9. **Comment adverbs** makes comments about what is being said, e.g. "he wisely didn't say a word".
10. **Linking adverbs** relates to a previous clause or sentence, e.g. "He worked very hard. However, he still had time to relax," and "in conclusion, we must invest internationally to survive".

Linguistic concept	IS modelling link
Adverbs	Adverbs relate, among other things, to the different aspects of the Zachman framework. For instance, "the client orders stock weekly/monthly" relates to the *when* aspect of Zachman's framework and "stock is stored in the stockroom" relates to the *where* aspect.

(f) Compound words

Words can occur in compounds. These compounds can occur with various combinations of lexical categories. For instance: noun + noun ("spaceship", "electronic mail"), adjective + adjective ("red-hot"), adjective + noun ("blackboard"), and noun + adjective ("earthbound", "pitch-black").

Linguistic concept	IS modelling link
Compound words	In many modelling situations, compound nouns are written as one word, for example "User_Rights" or "UserRights".

(g) Word relationships

It is also important to realise that specific words can be linked across lexical categories. For instance the verb "pay" is related to the nouns "payer" and "payee", and the adjective "payable", while the adverb "quickly" is linked to the adverb "quick".

Linguistic concept	IS modelling link
Word relationships	There are no specific uses of this concept in modelling.

(h) Conjunctions

Conjunctions connect words or groups of words, e.g. "you and I are a couple".

A conjunction is employed to connect words, phrases or clauses. For example, "he is fat and ugly", "we went to the movies after we had dinner" and "the kind and generous man gave alms to the poor".

A conjunction can belong to one of the following classes:

1. **Subordinate conjunctions** connect two finite clauses by making one clause subordinate to the other, e.g. "When I walked down the street, I saw him on the road".
2. **Coordinating conjunctions** connect words of the same formation and grammatical class, e.g. "John and Mary are married", "I will work and study next year".
3. **Correlative conjunctions** are pairs of conjunctions that behave together like subordinate conjunctions, e.g. "he neither works nor studies", "the more, the merrier" and both John and Susan are engineers".

Linguistic concept	**IS modelling link**
Conjunctions	Relates to Boolean logic in modelling and programming. It occurs mostly in conditional statements, e.g. *If the salary > x and number of years in the company > 20 then …*

(i) Interjections

Interjections express emotion, e.g. "Wow, what a concert!"

Linguistic concept	**IS modelling link**
Interjections	Not used in any IS modelling.

(j) Determiners

Determiners call attention to nouns by occurring before the nouns, e.g. "a mob damaged his bicycle". The most frequently used determiners are "a", "an", and "the". The determiner "the" makes the noun it determines definite, e.g. "The child fell

down" (a specific child). The determiners "a" and "an" make the noun they determine indefinite, e.g. "a child fell down" (any child). Articles occur with noun phrases and can either be definite ("the") or indefinite ("a", "an").

Linguistic concept	**IS modelling link**
Determiners	Not really used in IS modelling, for instance, the name of a use case would rather be "register new client" than "register the new client".

(k) Prepositions

Prepositions indicate a semantic relationship between entities, such as the following:

1. Location of one entity in relation to another, e.g. "the book is on/under/above/below/near the bookshelf".
2. Direction, e.g. "he travelled from his house to work".
3. Accompaniment, e.g. "with/without salt".

A preposition is one or more words that reveal the relationship between the object of the proposition and some other word in the clause. A preposition and its object constitute a prepositional phrase. For example, "the cost of this book is high".

Some prepositions relate to place, such as "in", "inside", "under", "across", "on top of", "below", "in front". Most prepositions of place indicate where something is or where it is going. For example, "there was a barrier across the road" (position) and "the man ran across the road" (movement). Prepositions of place can also have more abstract meanings, e.g. "I'm into classical music", "his behaviour is above reproach and "the people are behind their manager".

Some prepositions of place are one-dimensional. "At" is used when we see something as a point in space, e.g. "he was waiting at the house". Some are two-dimensional: "on" can be used for a surface, e.g. "the picture is on the wall", or it can be used for a

line, e.g. “the house is on the main road”. Some are three-dimensional: “in” is used when we see something as all around, e.g. “the man in the blue shirt”.

Prepositions can also indicate time, e.g. “we met in 1999”, “on Tuesday”, “in spring”, “during the week”, “since last week”.

There are many idiomatic phrases beginning with a preposition, e.g. “he drives at top speed”, “I saw it on television”, “we arrived in time for dinner”, “we arrived on time for dinner”, “we arrived in good time for dinner” and “we arrived just in time for dinner”.

Linguistic concept	**IS modelling link**
Prepositions	In modelling, the presence of prepositions indicates relationships, mostly spatial or time-related.

(l) Pronouns

Pronouns are words that are usually used in place of nouns or noun phrases, e.g. “she looked him in the eye”. The noun or noun phrase that is replaced by a pronoun is called the referent (or antecedent) of the pronoun. For example, in the sentence “Anita walked to the door where she saw her younger brother leaning on crutches; he was wearing a cast on his left foot”, the referent of the pronoun “she is Anita”, while the referent of the pronoun “he” is Anita’s younger brother.

The referent of a pronoun need not always be a particular noun or noun phrase. For example, in “he cheated, but it did not help him to succeed”, the referent of “it” is his cheating. The referent of a pronoun can often be found by seeing how the pronoun is declined.

Linguistic concept	**IS modelling link**
Pronouns	Pronouns are never used in modelling. The noun to which the pronoun refers is normally used.

5.2.2 Syntax

The study of the structure of sentences is called syntax. Sentences are made up of clauses, clauses are built up from phrases and phrases are built up from one or more words.

5.2.2.1 Phrases

There are five kinds of phrases:

1. **Verb phrases** consist of an ordinary verb ("come", "sing") plus optional auxiliary verbs ("is", "had", "can").
2. **Noun phrases** consist of a noun and usually a determiner in front of it. A noun phrase can also be a pronoun. A **noun phrase** is a group of words that is not a clause but, as a unit, behaves like a noun. For example, "the rowdy boys were punished". The word "boys" is the vital part of the noun phrase. It is called the *headword* of the noun phrase.
3. **Adjective phrases** consist of an adjective, sometimes with an adverb of degree ("very").
4. **Adverb phrases** consist of an adverb, sometimes with an adverb of degree ("almost").
5. **Prepositional phrases** consist of a preposition plus a noun phrase.

Linguistic concept	**IS modelling link**
Noun phrase	Composite noun phrases are normally translated into a single name, such as "EmployeeLeave" or "Employee_Leave".

5.2.2.2 Clauses

Sentences are made up of one or more main clauses. A main clause has a finite verb. "And", "or", "but", and "so" are used to join main clauses, e.g. "it was late and I was tired". A subclause is part of a main clause, e.g. "the wind caught him as he fell",

"I was tired because I was working". We use "because", "when", "if", "that", etc. in subclauses.

Clauses are built up from phrases. The elements of an English clause are as follows:

- **Subject:** The person or thing about which the clause is.
- **Predicate or verb:** It describes what the subject did, what action was done to the subject or what state of existence the subject is in.
- **Object:** This is a person or thing affected by the action of the verb.
- **Complement:** This relates to the subject.
- **Adverbial:** This relates to the verb.

The **subject** of a sentence is those words that tell us what the sentence is about. If the subject of a sentence comprises more than one part (connected by the words "and", "but" or "or") the subject is a **compound subject**. For example, "Jan and Susan helped with the chores".

The **predicate** is those words that do not constitute the subject. The predicate of a sentence tells us the following:

- What the subject did: "Susan toured France".
- What action was done to the subject: "Susan was cheated by the guide".
- What state of existence the subject is in: "Susan looked ill".

Normally the subject occurs before (to the left) of the predicate, but they can be transposed. For example, "*Ill looked Susan*".

When the predicate explains more than one action or more than one state of existence (connected by the words "and", "but" or "or"), then the predicate is a **compound predicate.** For example, "she hopped, skipped and jumped". Note that the sentence "he ate curry and rice" is not a compound predicate, because it explains only one action, namely "eating".

Sentences can have both a compound subject and compound predicate.

A group of words containing a subject and predicate constitutes a **finite clause.** A sentence has at least one finite clause. A **non-finite clause** is a group of words that express some sense of action or a state of existence, but the clause can never exists by itself, and is connected to some finite clause. For example, "I appreciated his visiting me".

Some basic clause patterns are as follows:

- *A train – stopped* (subject – intransitive verb)
- *Five men – carried – the bag* (subject – transitive verb – object)
- *The student – was – unlucky* (subject – verb – complement)
- *A course – is presented – every semester* (subject – verb – adverbial)
- *The mother – gave – the baby – its dummy* (subject – verb – indirect object – direct object)

Note that all clause patterns contain a subject and a verb in that order. The most common clause pattern is subject – verb – object.

5.2.2.3 Sentence

A sentence is a grammatically autonomous word group that makes sense by expressing a thought. Sentences are used to make statements, ask questions and issue directions. Sentences can be simple, i.e. they consist of one clause that stands on its own, or complex, i.e. they consist of two or more clauses. A sentence can have positive or negative polarity (e.g. "she is there" vs. "she is not there").

Different kinds of sentences can be identified: Elliptical sentences are sentences from which words have been elided (deleted), for example, "(If) garbage (goes) in, (then) garbage (comes) out". Existential sentences are sentences containing an expletive like "there", for example, "there are several lamps on the stand" is equivalent to "several lamps are on the stand". Declarative sentences make a statement and ends with a

period. Imperative sentences issue a command or a request. The subject, usually "you", is often elided. For example, "please lend me the book". Interrogative sentences ask a question and ends with a question mark. The subject is the same as the corresponding declarative sentence. Exclamatory sentences express emotion and end with an exclamation mark. They can also take on the structure of a declarative, imperative or interrogative sentence. For example, "Isn't she lovely!"

A sentence S constitutes a noun phrase NP followed by a verb phrase VP. This can be indicated as follows:

S →NP VP

A noun phrase can have different formats, such as:

NP →N	(N = Noun)
NP → DET N	(D = Determiner)
NP → DET ADJ N	(ADJ = Adjective)

A verb phrase can have different formats, such as:

VP → V	(V = Verb)
VP → V NP	
VP → V NP PP	(PP = Prepositional phrase)

Linguistic concept	**IS modelling link**
Sentence	Many modelling constructs can be translated into sentences. For instance, a use case diagram can be translated into a sentence by making the actor the subject, and the use case name the predicate and object, such as "client orders product".
Sentence polarity	Related to Boolean logic. Appears mostly in a conditional statement like "if it is not the end of the month, then…".

Linguistic concept	IS modelling link
Elliptical sentences	Not used in modelling because everything must be explicitly stated.
Existential sentences	Not normally used in modelling.
Declarative sentences	Most statements in modelling can be translated into declarative sentences.
Imperative sentences	Used when modelling instructions to a user of a system. For example, “place the paper in the printer”. Normally used in business rules and conditions and constraints.
Interrogative sentences	Not really used in modelling.
Exclamatory sentences	Not really used in modelling.

5.2.3 Semantics

Semantics is concerned with meaning on both word and sentence level. Three types of meaning can be identified: referential, social and affective meaning.

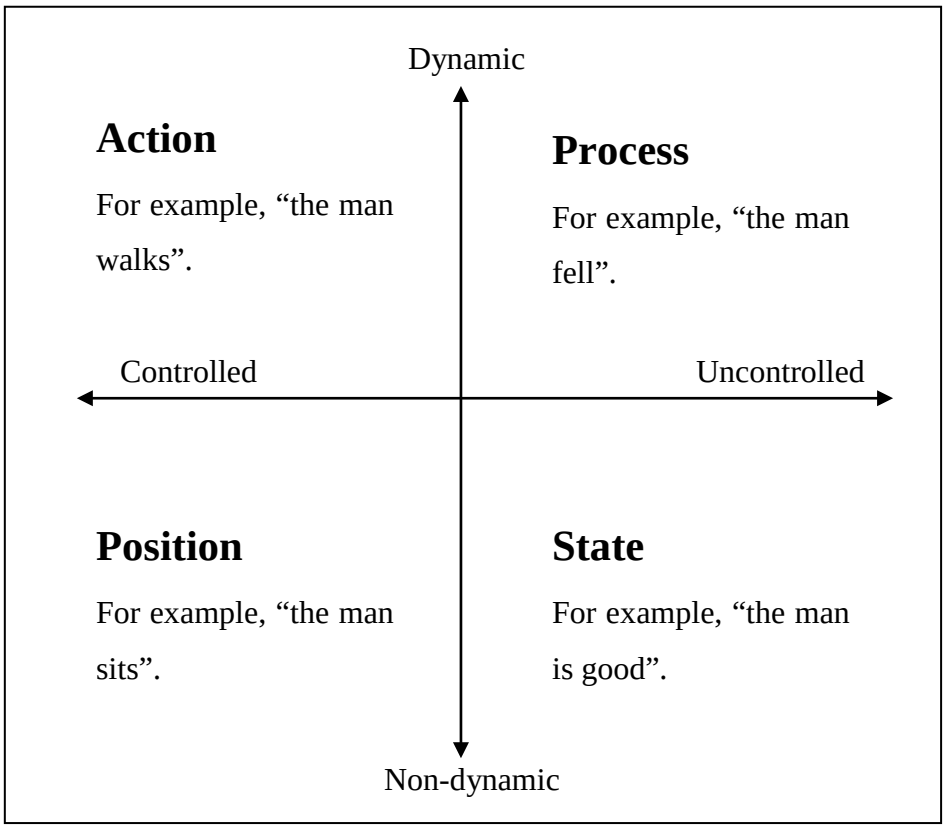
Referential meaning refers to looking for the meaning of a word or sentence by considering the person, object, abstract notion, state or event to which the word or sentence refers. In a referring expression, like “John’s car”, the specific car belonging to John is the **referent** of the expression.

Social meaning refers to the fact that over and above referential meaning, the choice of words can also convey social class, ethnicity, regional origin, gender and context. For instance, people calling their drink “pint”, “beer” or “lager” can indicate different social classes.

With **affective meaning,** the choice of words conveys the language user’s feelings, attitudes, and opinions. For example, by using the word “speed cop” instead of “traffic officer” a different level of respect is indicated.

5.2.3.1 Semantic functions

The Functional Grammar of SC Dik distinguishes between four states of affairs (or predications), based on the parameters controlled/uncontrolled and dynamic/non-dynamic (Kroeze, 2008). These four predications are summarised in Table 5-1.



Adapted from Kroeze (2008)

Table 5-1: Predications

Linguistic concept	IS modelling link
The four states of predication: action, process, position, state	In IS, no distinction is normally made between the first three states of predication. All of them will be represented by either a use case or function or program or any other action-related construct. State, on the other hand, is specifically specified in especially UML, but also implicitly in ERDs.

A predication is the combination of the predicate plus compulsory terms (or arguments) and optional terms (or satellites). Arguments and satellites can have different semantic functions (roles which the referents of the terms fulfil in the predication) (Kroeze, 2008; Kroeze, 2003; Weigand, 1992).

The following are different semantic functions that can be identified (Kroeze, 2008; Dik, 1997a; Dik, 1997b):

- **Agent:** The controller of an action, e.g. "the dog chases the car".
- **Positioner:** The controller of a position, e.g. "he maintains the peace between the different negotiating parties.
- **Force:** The non-controlling entity that initiates a process, e.g. "the exchange rate fluctuation caused the stock prices to fall".
- **Processed:** The entity passively undergoing a process, e.g. "the average cost price slid to an all-time low".
- **Zero:** The entity primarily involved in a state, e.g. "the price is high" (the price does not control the state – it just happens to be in it).
- **Patient (or goal):** The patient is the entity affected or effected (produced) by the operation of some agent, positioner, force or processor, e.g. "the manager prints the report".
- **Receiver (or recipient):** The entity to which something is transferred as a possession, e.g. "the employer paid the salary to the employee".
- **Location:** The place where something is located or where a predication takes place, e.g. "the cashier works in the front office".
- **Direction:** The entity towards which something moves or is moved, e.g. "They sent the order to the Procurement Department."
- **Source:** The entity from which something moves or is moved, e.g. "The supplier mails the invoice from the factory".
- **Reference:** The second or third term of a relation with reference to which the relation holds, e.g. "The policy reflects the company's mission statement".
- **Interested party (or beneficiary):** The person or institution to the advantage/disadvantage of whom the predication is effected, e.g. "The strategic report is produced for top management."

- **Company:** The entity together with which the predication is effected, e.g. "Finance created the feasibility study together with the IT Department".
- **Instrument:** The tool with which an action is executed or with which a position is maintained, e.g. "The credit clerk determines the client's credit rating by means of the credit rating procedure."
- **Manner:** The way or manner in which an action is executed, a position is maintained or a process takes place, e.g. "The developer creates the program according to the company's development standards".
- **Speed:** Indicates the quantity of action or process which is run through per time unit, e.g. "The project needs to done 25% faster to reach the deadline".
- **Role (or quality):** The role/function/authority/capacity by virtue of which an action is executed or a position is maintained, e.g. "As the head of the department, Alta performs appraisals".
- **Path (or route):** Indicates the orientation or route of a movement, e.g. "The Finance Department sends the invoice via the supplier's standard ordering channel".
- **Time:** The time at/from/until which a predication takes place, e.g. "Financial year ends on 30 September".
    - **Duration:** A subcategory of time. The period of time in which a predication takes place, e.g. "The quote is valid for 5 days".
    - **Frequency:** A subcategory of time. The number of times that a predication is repeated in a certain period, e.g. "The start-of-day procedure must be executed every week day at 07:00".
- **Circumstance:** A second predication taking place at the same time as the main predication, e.g. "While the cake is in the oven, the icing can be made". Some subcategories of circumstance are as follows:
    - **Real condition:** E.g. "If the order is bigger than 20, give 10% discount".
    - **Unreal condition:** E.g. "If the profit is 500%, the company can pay off all its debt".
    - **Concession:** E.g. "Although a client is a pensioner, they get full benefits".
    - **Exception:** E.g. "The fee is R50, but children pay R10".
    - **Restriction:** E.g. "Projects greater than R10 million cannot be authorised without the steering committee's approval".

- **Result:** A second predication which is brought about as the result or consequence of the main predication, e.g. "When an order is placed**,** the stock levels are updated".
- **Purpose:** A second predication in the future the controller deliberately wishes to bring about by means of the main predication. The purpose serves as the motivation for the main predication, e.g. "The execution of the audit procedure will ensure compliance with the audit standards at financial year-end".
- **Reason:** A motivation for the occurrence of a controlled predication in terms of a causal ground ascribed to the controller, e.g. "The project team worked overtime, because the project manager required them to".
- **Cause:** A motivation that is not ascribed to any of the participants of the predication, but which is given by the speaker as an explanation for the occurrence of the predication, e.g. "The building project was late because of excessive rainfall".

Semantic functions expressed by non-verbal predicates (nouns, adjectives, adverbs and prepositional phrases) are as follows:

- **Existence:** An argument expressing the mere existence of a zero-argument, e.g. "Inflation will always be with us".
- **Identity:** An argument expressing the identity or species of the zero-argument, e.g. "The university alumni are students who have completed their degrees at the university".
- **Class:** An argument that designates the class of which the subject is a member, e.g. "He is a permanent member of staff".
- **Quality (or property assignment):** An argument expressing the quality of characteristics of the zero-assignment, e.g. "His age is 40 years".
- **Possessor:** A term indicating the owner of the zero-argument or other element, e.g. "The receiving branch becomes the owner of the rental car".

Linguistic concept	**IS modelling link**
Agent	The agent is normally the actor or external agent in various modelling techniques.

Linguistic concept	**IS modelling link**
Positioner	In IS modelling, a positioner is never explicitly distinguished.
Force	Not explicitly indicated but implicitly, for instance, when an actor has an “initiates” stereotype in use case modelling, the actor is the equivalent of a force.
Processed	In IS modelling a processed is never explicitly distinguished.
Zero	An entity in ERDs and a class in UML is a zero in relation to their respective attributes.
Patient and receiver	A patient is never explicitly distinguished in IS modelling. A receiver can be indicated in use case modelling as an external receiver actor (ERA).
Location	Location is rarely indicated in IS modelling and then only implicitly as in the deployment diagram in UML.
Direction and source	A direction and source are never explicitly distinguished in IS modelling.
Reference	Not used in IS modelling.
Interested party	An interested party is not separately distinguished, but is included in, for instance, use cases as an actor.
Company	Not used in IS modelling.
Instrument	Not directly used in IS modelling, but relates to the means or mechanism of a process in IDEF0.
Manner	Not used in IS modelling.
Speed	Not used in IS modelling.
Role	This concept is used a lot in IS modelling, but no specific role modelling construct exists.
Path	Not used in IS modelling.
Time, duration, frequency	Very important concept in modelling, but it is only really in UML that time is explicitly modelled.
Circumstance	Relates to concurrent activities, as modelled in a UML activity diagram.

Linguistic concept	IS modelling link
Real condition, unreal condition, concession, exception, restriction	These are all related to business rules, but no specific modelling constructs exist for them, except secondary ones like a decision symbol in a UML activity diagram.
Result	Relates to the output of an IDEF0 diagram.
Purpose, reason, cause	They all relate to the "why" aspect of Zachman, but no specific modelling construct exists.
Existence	Relates to the associations between entities in ERDs and classes in UML.
Identity	Relates to the definition of a term.
Class	Relates to the inheritance or generalisation/specialisation concept in OO.
Quality	Relates to the values of the attributes of an entity (ERD) or a class (UML).
Possessor	No explicit construct in IS modelling, but could be related to the owner in a CATWOE table (soft system methodology).

5.2.3.2 Lexical semantics

Lexical semantics is concerned with the relationships among word meanings (Stabler, 2010; Shinghal, 1992; Valeika and Buitkiene, 2003; Kornai, 2007; Haspelmath, 2001).

(a) Hyponymy

A hyponym is a term whose referent is totally included in the referent of another term, for instance, "blue", "red" and "yellow" are all hyponyms of "colour". The "higher" term, "colour", is called the hypernym. Hyponymy is not restricted to nouns or adjectives only, but can also occur with verbs and other grammatical classes, for instance, "walk" can be the hypernym for "stroll", "saunter", "amble", "hike", etc.

Hyponymy can exist at more than one level, for instance, “aquamarine” and “royal blue” are hyponyms of “blue”, which is a hyponym of “colour” in turn.

Linguistic concept	**IS modelling link**
Hyponymy	An extremely important concept in modelling which relates to “a kind of”, “is-a” relationship or inheritance relationship in OO.

(b) Part-whole relationships

Part-whole relationships are where the referent of one term is included in the referent of the second term, for instance, “room” and “house”. It differs from hyponymy in that a room is not a type of house, but in (part of) the house.

Linguistic concept	**IS modelling link**
Part-whole relationship	Relates to the aggregation concept in OO.

(c) Synonymy

Two words are synonymous when they mean the same thing. More formally, when every referent of term A is a referent of term B, and vice versa. For example, “rent” and “hire” can be synonyms.

Linguistic concept	**IS modelling link**
Synonymy	Important concept, especially in analysis. Most methodologies indicate the importance of identifying synonyms when defining terms. Mostly, one term will be seen as the main term and all other as synonyms of that term.

(d) Antonymy

Antonymy denotes opposition in meaning and is a binary relationship, unlike synonymy and hyponymy. The most obvious examples are pairs of adjectives that describe opposite concepts, such as “hot” and “cold”, “open” and “closed”, “dead” and “alive”. However, nouns like “male” and “female”, adverbs like “always” and “never”, and verbs like “love” and “hate” are also antonymous.

There are different kinds of antonymy. Words such as “large” and “small” are fairly subjective, e.g. a mouse is smaller than a house but much larger than a virus. These pairs are called **gradable**. Typically, for gradable antonyms there are words or expressions to describe intermediate words like “medium large”, and “fairly small”. In contrast, words like “single” and “married” are mutually exclusive and complementary. A person cannot be both at the same time. These pairs are called **non-gradable**.

Linguistic concept	**IS modelling link**
Antonymy	This concept does not feature directly in IS modelling.

(e) Converseness

Converseness refers to a reciprocal concept of oppositeness, different from antonymy. Take, for example, the words “husband” and “wife”. The word “husband” is the converse of “wife”, because if A is the husband of B, then B is the wife of A.

Linguistic concept	**IS modelling link**
Converseness	Does not feature directly in IS modelling.

(f) Polysemy and homonymy

When a word has more than one meaning, it is polysemic, e.g. “book” can be used as follows: “he reads a book” or “they book their flights”. When words sound the same but have different meanings they are homonymic, e.g. “there” and “their”.

Linguistic concept	IS modelling link
Polysemy	Does not feature directly in IS modelling. But ambiguous terms are normally clearly identified (in a list of terms) to indicate just one meaning.
Homonymy	Does not feature in IS modelling.

(g) Metaphorical extension

A metaphor is an extension in the use of a word beyond its primary meaning. It describes referents that are similar to the word's primary referent. For instance, the word "heart" can, over and above its primary meaning of the biological pump, also be used to describe the centre of an issue (the heart of the matter), the seat of emotion (she has broken his heart), etc.

Linguistic concept	IS modelling link
Metaphorical extension	Does not feature directly in IS modelling.

(h) Tense and modality

The semantic category **tense** indicates the time reference of a word or an entire clause.

Epistemic modality indicates the attitude speakers have towards the truth of the statements they make. For instance, "they are probably right" indicates probability, "they are right" indicates assertion and "they know what they are talking about, so they should be right" indicates conjecture.

Deontic modality expresses obligation, permission or suggestion. For instance, "he must wash the car" indicates command, "he may wash the car" indicates permission, while "he washes the car" indicate statement.

The modalities are related and the same auxiliary words (like “may”, “must” and “should”) can indicate both types. Modal verbs (like “order”, “allow”, “command” and “assume”) and modal adverbs (like “possibly”, “probably” and “certainly”) also indicate modality.

Linguistic concept	**IS modelling link**
Tense	Tense indicates mostly the as-is and the to-be situations in modelling.
Epistemic modality	Probability and conjecture are not normally taken into consideration in modelling. Only a few techniques like decision trees do allow for indicating probability, and rich pictures (in SSM) allow for conjecture. Mainstream modelling techniques cater mostly for assertion.
Deontic modality	Command normally indicates the presence of business rules or instructions to users.

(i) Reference

Reference provides information about noun phrases and their referents. For example, note the semantic difference between the following two sentences: “he reads the book” and “he reads a book”. The first assumes the speaker can identify the book, while the second doesn’t.

Linguistic concept	**IS modelling link**
Reference	When the referents can be identified, for instance, in the phrase “the product”, it indicates a cardinality of one, a singleton (OO) or an instance of an entity or class (in this case “product”). The phrase “a product”, on the other hand, indicates a cardinality of many and the class or entity itself.

(j) Deixis

Deixis identifies the orientation of objects or events in relation to specific points of reference. All types of deixis share a basic point of the reference: the speaker's identity and location in space and time.

Personal deixis shows the orientation of our communications with respect to ourselves, our conversational partners and third parties. These are mostly indicated by personal pronouns. First-person pronouns (such as "I", "we" and "us") refer to the speaker or group including the speaker. Second-person pronouns (like "you") refer to the addressee or group including the addressee. Third-person pronouns (like "he", "she", "it" and "they") indicate any other entity besides the speaker and person (or persons) spoken to. Depending on the language, gender, number and even social status can also be indicated.

Spatial deixis indicates in a language expression the spatial orientation of the referent of an action or state. Spatial deixis are mostly indicated by demonstratives (like "this" and "that"), adverbs of place ("here" and "there") and directional verbs (like "go", "come", "bring" and "take"). The main reference points are near or far from the speaker.

Temporal deixis indicates in a language expression the time orientation of the referent of action or state. The most basic orientation is the moment at which the expression is uttered. Events before that moment are in the past, during that moment are in the present and after that moment are in the future.

Linguistic concept	**IS modelling link**
Personal deixis	Personal deixis can indicate the presence and orientation of a business conversation or transaction. The interaction between a user and an ATM is a classic example: "user inserts card, ATM verifies card", etc. In IS modelling, the third-person perspective is mostly that of the system being modelled.

Linguistic concept	IS modelling link
Spatial deixis	Spatial deixis indicates the "where" aspect of Zachman, as well as actions involving the movement of either physical or informational entities, for instance, "the clerk emails the invoice to the client".
Temporal deixis	Relates to the "when" aspect of Zachman.

5.2.4 Pragmatics

For a long time, linguists studied individual sentences in isolation. But language is normally used in larger units, like conversations, monologues, emails or letters. These larger units are studied in pragmatics (also called information structure).

In any sequence of sentences, speakers and writers will mark some elements as more important (highlighting) or less important (backgrounding). This is called **information structure** and takes into account the discourse context of a sentence.

(a) Discourse

A **discourse** is a series of sentences (or other non-verbal forms of communication) that go together, for example, a conversation in the tea room, an email, a television interview, a comment to you about the neighbour walking by, a speeding fine or telling a joke. These discourses are social instruments used for communication. Discourse can have a major effect on the structure of a given sentence.

A conversation is a discourse where more than one person is involved. Some of the properties of a conversation are as follows:

- Any reasonable number of people can take part.
- There are rules governing how people take turns.
- There are principles of socially acceptable conversation behaviour like greeting (opening the conversation) and closing the conversation.

Linguistic concept	IS modelling link
Discourse	Any interaction of a user with a system can be seen as a discourse or conversation. You have an opening (like logging in), a closing (like logging out) and all of the steps in-between.

(b) Topic

The main discourse function of the subject is to identify the **topic** or theme of the discourse. Topics represent **given information** – information the speaker assumes the hearer already knows. A topic only becomes a topic once it is introduced into the discourse. Once a topic is introduced, it stays the subject of subsequent sentences until a new topic is introduced. The topic is in contrast to the **comment**, the element that says something about the topic. The topic is not necessarily derived from a sentence, but can be derived from the discourse context, for example, "look how cute" when the speaker passes a baby in the street identifies the topic "the baby".

The context can be on different levels. Firstly, it can be linguistic, the utterances in the discourse preceding the current point. Secondly, it can also be the immediate physical or social environment. Thirdly, it can include general knowledge.

Linguistic concept	IS modelling link
Topic	When doing an analysis, it is important for the analyst to determine the topic when statements are made by users. For instance, the statement "the clerk validates the order", although syntactically and semantically valid, is incomplete pragmatically and must be placed within the topic of the "order process" along with all the other order process steps.

(c) Speech acts

Certain utterances only declare or state, but there are utterances that in the right circumstances perform an action. For instance, when the bride and bridegroom say "I

do", it constitutes entering into a legal contract; or when the supplier states "I will deliver this before 12:00 tomorrow", it constitutes a service contract. These types of utterances are called performative utterances.

There are four categories of speech acts (Searle, 1976):

1. **Utterance acts** are simply acts of uttering sound, words, phrases or sentences in a language and can be performed by a non-communicating entity like a parrot or tape recorder.
2. **Illocutionary acts** are acts performed in saying something. Examples of illocutionary acts are asserting, reporting, stating, asking, suggesting, ordering and proposing.
3. **Perlocutionary acts** are acts performed by saying something. Examples of perlocutionary acts are inspiring, persuading, intimidating, misleading, embarrassing and irritating.
4. **Propositional acts** refer to something and characterises it with a predicate. For instance, "the earth is flat" or "nobody is perfect".

Linguistic concept	**IS modelling link**
Speech acts	The sending of business transaction messages between different agents constitutes illocutionary acts. This forms a major part of IS and organisations.

5.3 Conclusion

Because language is so fundamental to modelling, linguistics is a very important reference discipline for modelling. Of special interest are linguistic concepts and constructs that are absent or underemphasised in modelling. Based on the comparison of linguistics and IS modelling in this chapter, some very interesting conclusions can be made.

One of the first insights is that linguistics makes a clear distinction between the different levels of morphology, syntax, semantics and pragmatics. In IS modelling,

some of these levels are often neglected. IS modelling, and the teaching thereof, mostly concentrates on the morphological, and to some degree the syntactical level, but not really on the semantic and pragmatic levels. For instance, when learning a new modelling technique like use case diagrams, the basic constructs such as agent, use case and association will be taught and examples given, but very few rules will be given of what a good use case "sentence" or "clause" is.

On a morphological level, the main conclusion is that only a subset of all the words used in a specific universe of discourse will actually be modelled. In a sense, the root meanings of words are used rather than derivations of those root words.

On a lexical level, a number of significant conclusions can be made:

- In language the two main things that are communicated are "things"[2] as represented by nouns and noun phrases and the relationships between them as represented by various other linguistic concepts. One of the most important relationships is that of action represented mostly by verbs. For instance, when somebody says Humpty-Dumpty sat on the wall, the relationship between the two things (Humpty-Dumpty and the wall) is indicated by the phrase "sat on". It mainly shows the spatial relationship between them, one under and the other on top. It also shows that this spatial relationship is not necessarily true now, but that it was true somewhere in the past, because it says "sat" and not "sits" or "is sitting".
- By contrast, in modelling the two main things that are modelled are "things" (agents, actors, entities, objects, etc.) and actions (processes, functions, programs, use cases, etc.), with the relationships between things taking at most a third place.
- Various lexical types give rise to a number of relationship types between things:
  a) **Action relationships** indicate dynamic relationships where subject things execute actions on object things in a finite (even if long) amount of time. The linguistic concepts indicating action relationships are action verbs, predicates

[2] The words "thing" or "things" is used rather than "object" or "entity" because of the current IT connotations of those words.

and prepositions. Action relationships can indicate many subtypes, such as the following:

- **Association relationships,** e.g. “Customer orders product”.
- **Movement** between two locations, e.g. “The flight transports passengers between origin airport and destination airport”.

b) Action and existence verbs also indicate **static relationships** showing permanent relationships between “things”, e.g. “Order consists of products” and “The customer order is filled”.

On a semantic level, semantic functions can be linked to many concepts in IS modelling, such as agent, role, etc. The interesting part is the big number of semantic functions that are either not explicitly defined or not defined at all in IS modelling. These can provide the basis for developing richer, more nuanced modelling constructs.

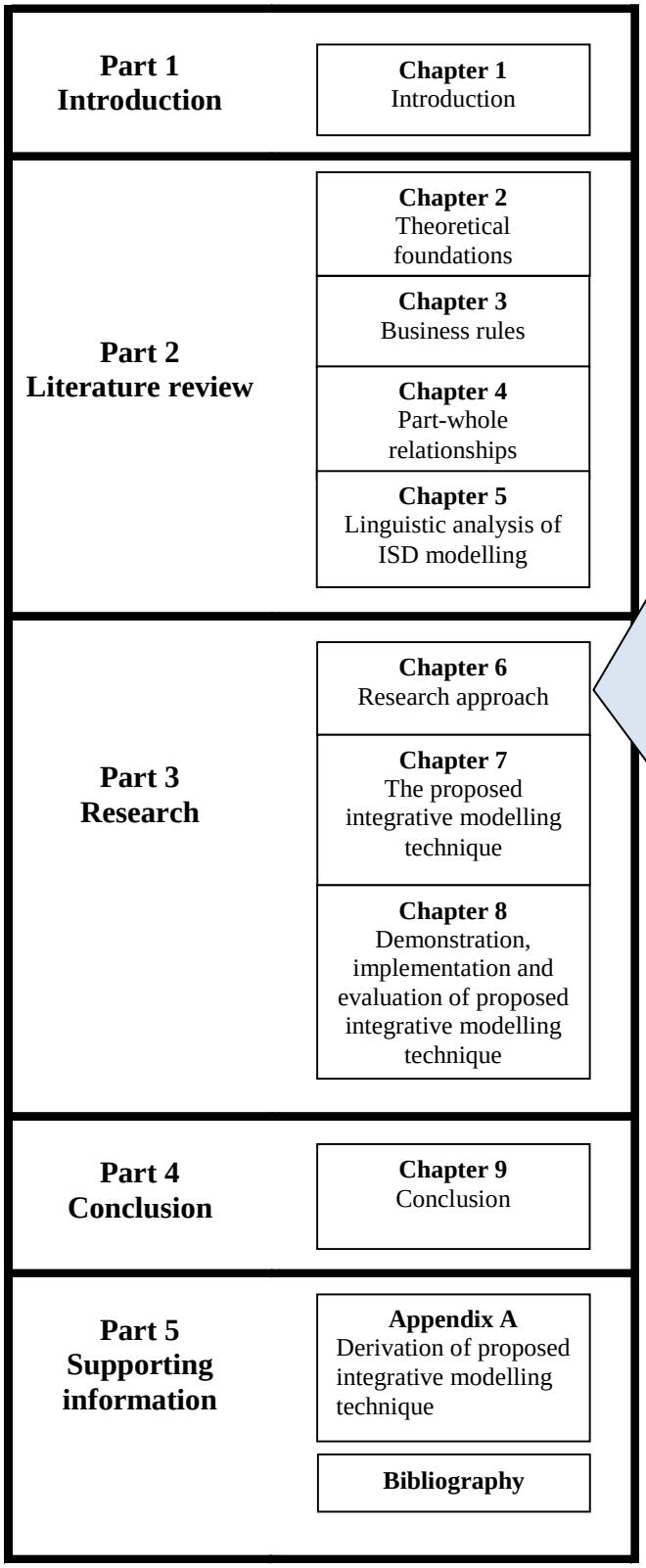
Further, on a semantic level, lexical semantics indicates relationships between words, many of which are present in IS modelling, like “inheritance” (hyponymy) and “aggregation” (part-whole relationships). However, the relationships missing from ISD modelling point to interesting opportunities to enrich modelling. For instance, converseness can help to identify processes or functions such as “buy” and “sell”, “input” and “output”, or “debit” and “credit”. By understanding that these functions go in pairs, finding one of the pair can cause an automatic query concerning the other half of the pair.

On a pragmatic level, it is clear that communication is not made up of loose sentences but of sentences structured together in bigger units forming discourses. Similarly, a series of IS modelling diagrams does not constitute a proper model of a specific universe of discourse. Modelling is only complete when all diagrams are properly placed within an integrated structure and related to a wider context encompassing the total IS under discussion.

Part 3

Research

6. Research approach



6.1 Introduction
6.2 Research objectives, statement and questions
6.3 Research design
- 6.3.1 Research philosophy
- 6.3.2 The grounded approach
- 6.3.3 Data collection

6.4 Design science research
- 6.4.1 Background
- 6.4.2 Research methodology
- 6.4.3 Research output
- 6.4.4 Research guidelines
- 6.4.5 Design science research theory

6.5 The journey
- 6.5.1 Part 1 – Grounded approach
- 6.5.2 Part 2 – Design science research

6.6 Conclusion

6.1 Introduction

The research approach used in this study is design science research. It basically involves building an artefact to solve some a problem situation, and then evaluating this artefact. Design science research has a goal of utility, while behavioural science research has the goal of truth. According to Hevner et al. (2004:98), “while it can be argued that utility relies on truth, the discovery of truth may lag the application of its utility”. According to Bajaj et al. (2005), systems analysis and design touches on many areas of design research.

The design science research process is dependent on a knowledge base of prior IS research as well as results from reference disciplines (Hevner and March, 2003). The data analysis techniques of grounded theory and the fundamental structure of linguistics were used as the main contributors to the knowledge base of this study, forming its kernel theories.

In this chapter, the general research approach is discussed, an overview of the grounded and the design science research approaches is given, the research process followed is explained and the data studied is described in more detail.

6.2 Research objectives, statement and questions

As a result of the problems in integrating business and ISD modelling as described in Chapter 1, the objective of this study is to develop an integrative technique between business and ISD modelling.

To achieve this objective, the fundamental theoretical foundations of business and ISD modelling need to be investigated. This study will do so, firstly, by trying to find the “what”, i.e. the basic, fundamental properties, parameters and dimensions of IS modelling and their relationships. More specifically, it aims to find the fundamental ontological constructs of business and ISD modelling. In other words, what are the fundamental “things” in IS and organisations that we should model? This will be done by using a grounded approach.

Secondly, the resultant modelling constructs and their relationships will be used to form an integrative modelling technique. This modelling technique will form the "artefact" that is the basis for the subsequent design science research study.

6.3 Research design

6.3.1 Research philosophy

There has been a long-standing debate in the IS field between the positivist and interpretivist traditions of research. Positivism (more generally empiricism) has historically been the major philosophical underpinning for IS research and systems development, especially in the USA (Mingers, 2004c). The assumptions of positivism are an unproblematic, rational and mechanistic approach to IS. The research methods used by positivist researchers are mostly quantitative (Hughes and Howcroft, 2000).

There is, however, an increasing appreciation for the fundamentally social nature of IS. This has led to an increase in research that focuses on human interpretations and meaning; in other words, interpretive research. Interpretivist researchers mostly favour qualitative research methods but also use quantitative methods (Hughes and Howcroft, 2000).

Following on this, Hevner and March (2003:111) consider IS research to adhere to two complementary paradigms: the behavioural science paradigm viewing IS as a social science and the design science paradigm viewing IS as a technical science: "the sciences of the artificial."

Other approaches have also come to the fore: critical theory, postmodernism and actor-network theory. Although there are many proponents for all of the above positions, there are also extensive criticism against all of these positions (Mingers, 2004a).

Deluca, Gallivan and Kock (2008) consider the issues in research and between various research paradigms to be based on the following four dialectics:

- The rigour vs. relevance objective
- Positivist vs. interpretive epistemology
- Quantitative vs. qualitative methodology
- Confirmatory vs. disconfirmatory evidence

Mora et al. (2007:3) summarise the four main research paradigms using a systems approach as follows:

P.1 The hard/functionalist/positivist systems approach: "The intelligible world is an organised complexity comprised of a variety of natural, man-made, and social systems that own a real existence."

P.2 The soft/interpretative systems approach (rejects P.1): "…the intelligible world can be studied freely through systemic lenses and under an intersubjective social construction…"

P.3 The critical/emancipative systems approach (neutral to P.1, rejects P.2): "…the intelligible world can be uniquely understood when it is studied freely from restrictive social human relationships and a variety of theoretically coherent systemic lenses are used…"

P.4 The critical realism systems approach (includes P.1 to P.3 as well): "…the world is intelligible for human beings because of its stratified hierarchy of organised complexities – the widest container is the **real domain** that comprises a multi-strata of natural, man-made and social structures as well as of event-generative processes that are manifested in the **actual domain** that in turn contains to the **empirical** domain where the generated events can or cannot be detected…"

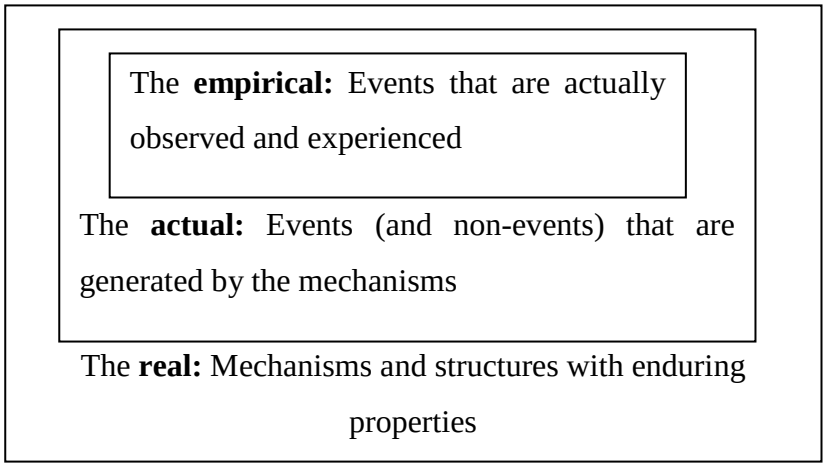
Mingers (2004a) proposes critical realism as an underpinning philosophy for IS, because it overcomes the criticism against the main IS philosophies, positivism and interpretivism. Critical realism in essence takes a realist view ontologically ("...there is an independently existing world of objects and structures that are causally active, giving rise to the actual events that do and do not occur"), a relativist view

epistemologically ("... our observations and knowledge can never be pure and unmediated, but are relative to our time period and culture...") and a retroduction (also called abduction in contrast to induction and deduction) view of scientific method ("We take some unexplained phenomenon that has been observed and propose hypothetical mechanisms that, if they existed, would generate or cause that which is to be explained.") (Mingers, 2004b:380, 385).

Dobson (2002) argues for using a critical realist position in IS research, because it elevates the philosophical issues, allows for more consistency in research and bridges the dualism between subjective and objective views of reality. Mingers (2004b) similarly considers critical realism as highly appropriate for IS, because it takes a fundamentally realist position (which the majority of IS academics and practitioners intuitively take), it addresses both natural and social sciences, therefore also including hard, soft and critical approaches, and it fits well with IS as an applied discipline.

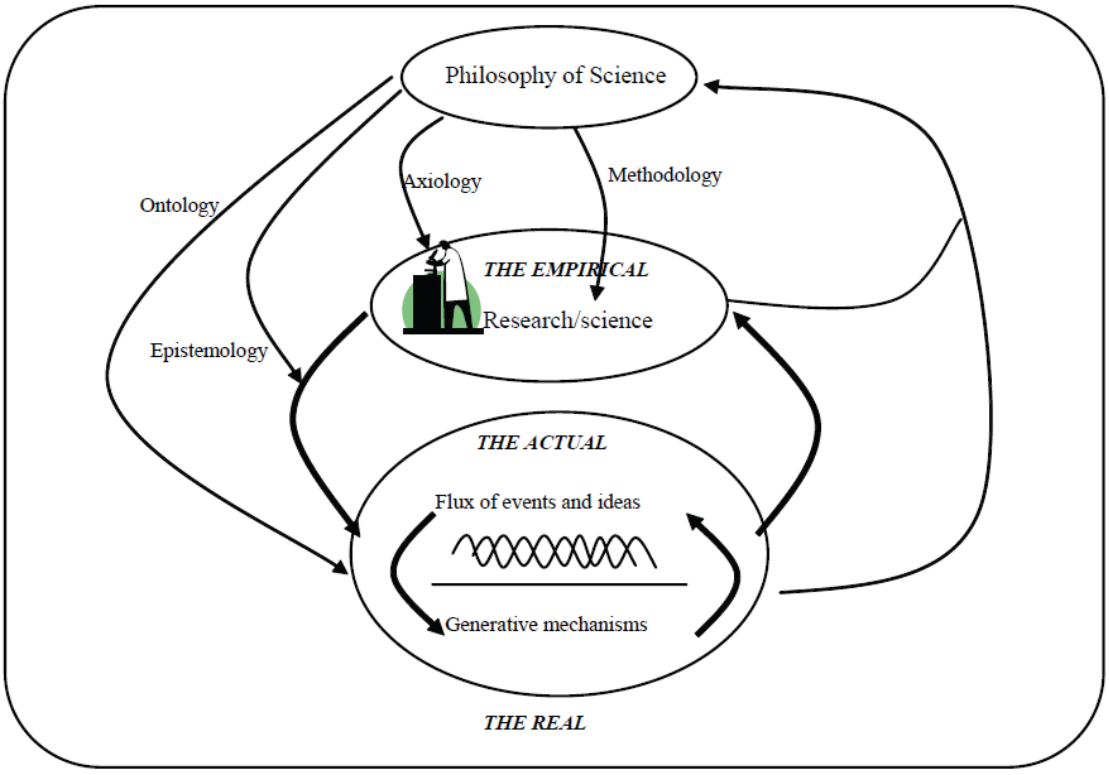
Bhaskar, one of the developers of critical realism, considers reality as intransitive (existing independently of humans) and stratified into different domains. The domains are the real, the actual and the empirical (see Figure 6-1 and 6-2 for the domains of science and philosophy with respect to the real, actual and empirical) (Mingers, 2004c):

- The **real:** Mechanisms and structures with enduring properties. This is what this study is trying to understand better with regard to ISD modelling. Mostly implicitly and sometimes explicitly, all ISD techniques assume some sort of *real*.
- The **actual:** Events (and non-events) that are generated by the mechanisms.
- The **empirical:** Events that are actually observed and experienced. Because the *real* was incorrectly or only partially identified, many of the ISD techniques developed over the years did not satisfy the modelling requirements of all the stakeholders, as the techniques did not correctly model the *actual* or the *empirical*. Therefore, there is the constant need to develop new techniques.



Mingers (2004c)

Figure 6-1: The three domains of real



Mingers (2008)

Figure 6-2: The domains of science and philosophy with respect to the real, the actual and the empirical

Critical realism does not prescribe a single research approach; it rather prescribes a certain attitude. Firstly, it never just describes, either qualitatively or quantitatively. It also attempts to gain an understanding of and explain things by considering the structures and mechanisms that affect observable events. Secondly, it recognises

different objects of knowledge (material, conceptual, social and psychological) and their holistic interactions with each type of object requiring a different research method. Thirdly, it realises the fallibility of observation (Mingers, 2004c).

In terms of design science theory, Gregor and Jones (2007) agree with Mingers and Bhaskar on critical realism as a valid approach to IS research, although they realise that there is ongoing debate in the field.

In terms of critical realism, the design science research artefact can be seen as an embodiment of the underlying structures and mechanisms that can empirically be evaluated, at least in terms of utility, but also to some extent in terms of truth. Therefore the purpose of this study, in terms of critical realism, is to find the underlying **mechanisms and structures** of business and ISD modelling, using the grounded approach, and then to embody these mechanisms and structures into a modelling technique that can be tested and evaluated **empirically** using design science research.

Research in IS using an inclusive methodological approach (using more than one methodology, as is the case in this study), rather than an exclusive approach, is advised by Davison and Martinsons (2011) for the following reasons:

- Research constrained to one methodology will be poorer in the improvements that it can generate on organisational reality.
- Research will make a bigger academic and business contribution when methods from various epistemological viewpoints are included.

Becker and Niehaves (2007) agree, but warn that the epistemological assumptions made by different researchers may differ greatly.

6.3.2 The grounded approach

The first part of this research used a grounded approach to determine the fundamental constructs of business and ISD modelling and their relationships, which are used to develop the design science research artefact. The grounded approach uses the data analysis techniques of grounded theory.

Grounded theory was developed by two sociologists, Glaser and Strauss (1967), while conducting an observational field study of how hospital staff dealt with dying patients. The purpose of grounded theory research is to organise the ideas coming out of an analysis of data, through a thorough analysis of documents, interview notes or field notes by continually coding and comparing data to produce a well-constructed theory (Parker and Roffey, 1997; Glaser, 2003).

The approach facilitates understanding to be fashioned into conceptual categories and concepts and then into theories or models. This is done without starting from an a priori definition. These concepts and theories are built from the "socially constructed knowledge of participants" (Daengbuppha, Hemmington and Wilkes, 2006:369; Hughes and Howcroft, 2000). Grounded theory differs from the other interpretive methodologies in a number of ways. It allows for the investigation of one or many cases. It does not only describe the subjects and their interactions, but tries to also develop a theory (Parker and Roffey, 1997).

The research does not start with a theory that must be verified. It starts with an area of study where theoretical constructs emerge from the study process. The implication is that data collection, analysis and the resulting theory have a reciprocal relationship. The research is trying to make sense of the data collected and giving it structure. It aims to generate theory by a three-phase process of induction, deduction and verification (Parker and Roffey, 1997).

The source of data is not only human interaction, but also includes interviews, written reports, minutes of meetings and other documents. It allows for the investigation of one or many cases. It does not only describe the subjects and their interactions, but tries to also develop a theory (Parker and Roffey, 1997; Goulding, 1998).

Grounded theory has diversified since its beginnings, with the most important variation between Glaser and Strauss on assumptions and methods. Glaser kept to the original theory, while Strauss, together with Corbin, reformulated the original theory (Heath and Cowley, 2004).

A first difference revolves around the issue of keeping research free from past experience and reading. Glaser (1978) sees the key process as induction, where the researcher moves from data to empirical generalisation to theory. Strauss and Corbin (1998), on the other hand, stress deduction and verification.

A second difference is related to coding procedures leading either to theory construction or theory discovery. Glaser and Strauss originally identified two levels of coding: firstly, coding into as many categories as possible, and secondly, coding an integration of the categories. Strauss and Corbin (1998) moved away from that position, as described in Table 6-1.

	Strauss and Corbin	**Glaser**
Initial coding	*Open coding* Use of analytic technique	*Substantive coding* Data-dependent
Intermediate phase	*Axial coding* Reduction and clustering of categories (paradigm model)	• Continuous with previous phase • Comparisons, with focus on data, became more abstract, categories refitted, emerging frameworks
Final development	*Selective coding* Detailed development of categories, selection of core, integration of categories	*Theoretical* Refitting and refinement of categories which integrate around emerging core
Theory	Detailed and dense process fully described	Parsimony, scope and modifiability

Heath and Cowley (2004)

Table 6-1: Data analysis: Glaser and Strauss compared

This study initially followed the Strauss and Corbin approach, but later on the Glaser approach. This process is described in section 6.4.

With regard to the use of literature, the theory that develops, guides the researcher to the literature that best informs, explains and contextualises the findings (Goulding, 1998).

6.3.3 Data collection

As stated before in section 1.3.1, Iivari et al. (2001a) created a hierarchical four-tier framework in an attempt to categorise all ISD-related issues: ISD paradigms, approaches, methods and methodologies, and techniques.

Out of these four areas, ISD techniques were used as the basis for data collection. The reason for that is the techniques in use today are the practical embodiment of the paradigms, approaches and methodologies that support them. In this sense, techniques are more fundamental and basic than the other aspects.

All the research done on ISD, as well as all the industry experiences, can be seen as conversations on ISD modelling. Formalised ISD techniques have in a sense captured these conversations in more real ways than ISD paradigms, approaches or even methodologies. When people advocate a specific ISD technique, they make a number of assumptions about how they see the underlying ISD reality that must be modelled.

In this study, these “conversations”, as captured in ISD techniques, is the “text” or data for the study. The main techniques of the past, the main techniques that are currently popular as well as lesser-known techniques were chosen to give as broad a range of techniques as possible.

This study could have done data collection by having interviews with people working with ISD techniques in many organisations, by observing them during the process of using ISD techniques, or by gathering the documents created as a result of these ISD techniques. All of these would have been valid and very useful, but they would have been limited and would not have taken the broader view. It was thus decided to rather conduct “interviews” with authors explaining the various ISD techniques to others in books and articles. These descriptions are normally the result of a lot of academic research and practical experience distilled into a set of specific techniques. The work done on these techniques have influenced and changed ISD and can be seen as conversations and interactions between various stakeholders (Wolfswinkel, Furtmueller and Wilderom, 2012). Myers (2009:107) supports this view by stating that if you only want to use grounded theory for “ ... your qualitative data analysis,

and some other theory as an overarching framework for your study, then I believe that is acceptable."

The ISD techniques studied (extended from Avison and Fitzgerald, 2003) as well as the specific sources used to represent each one of the techniques as part of the grounded approach used, are given in Table 6-2.

The basic principle in the selection of the sources was to use generally accepted, internationally recognised standards, like the IDEF0 standard. If there were no such standard, the most authoritative source on the relevant techniques was used, e.g. Chen (1976) for data modelling.

Techniques	Source
1. **Holistic techniques:** The soft system methodology (SSM) is used to represent the holistic approach, emphasising techniques such as rich pictures, root definitions (CATWOE) and conceptual models.	• The article by Checkland where he takes a thirty-year retrospective view of SSM (Checkland, 2000).
2. **Data techniques:** Entity modelling and structured query language (SQL) are used to represent the data approach to ISD.	• The seminal article by Chen explaining entity relationship modelling for the first time (Chen, 1976). • Microsoft SQL Server Books online – notes on SQL.
3. **Business process techniques:** The techniques to represent processes are many and can be categorised as follows: **Functional modelling:** IDEF0 and IDEF3 **Information modelling:** DFD, IDEF1 and IDEF1x **Dynamic modelling:** IDEF2, Petri-Nets, role activity diagram (RAD), agent relationship modelling (ARM) and agent/object life cycles (ALCs/OLCs). **Integrated modelling:** BPMN	• Certain techniques that fall under the process approach, like action diagrams and entity life cycles, will rather be studied via their object-oriented counterparts. • For functional modelling, the IDEF0 standard will be used (IDEF0, 1993) as well as other sources (Kappes, 1997). • Information modelling will be handled under data modelling. • Dynamic modelling will be represented by role activity diagrams, (RAD), as in Bal (1998), agent relationship modelling (ARM), as in Valiris and Glykas (2004), agent/object life cycles (ALCs/OLCs) as in Valiris and Glykas (2004), and agent-object relationship (AOR) modelling as in Wagner (2002). • Integrated modelling will be represented by BPMN as in OMG (2009).

Techniques	Source
4. **Object-oriented techniques:** The following techniques, as specified in UML, and grouped per view, are discussed: • The use case view of use case diagrams, and use case narratives • The static view of class diagrams and object diagrams • The dynamic view of sequence diagrams, collaboration diagrams and activity diagrams • The implementation view of component diagrams and deployment diagrams	• The UML 2 standard and other sources (OMG, 2007b; OMG, 2007a; France et al., 1998).
5. **Project management techniques:** Various project estimation techniques, PERT chart, Gantt chart and critical path method (CPM) techniques represent the project management approach.	• PMBOK notes on PERT, Gantt and CPM techniques (PMBOK, 1996).
6. **Organisational techniques:** Lateral thinking, critical success factors, scenario planning, future analysis, SWOT analysis, case-based reasoning, risk analysis.	Not studied because techniques do not describe specific ontological objects but rather processes.
7. **People techniques:** Stakeholder analysis, joint application design (JAD), joint requirements planning (JRP).	Not studied because techniques do not describe specific ontological objects but rather processes.
8. **Enterprise architecture techniques:** The main approach is the Zachman framework. The open distributed processing (ODP) standards are also discussed.	• The analysis of the Zachman framework for enterprise architecture from the GERAM perspective by Noran (2003). • The open distributed processing (ODP) standards (Toussaint, Baker and Groenewegen, 1997).
9. **Process logic description techniques:** Techniques describing process steps at a lower level of detail than processes including decision trees, decision tables, structured English, structure diagrams, Warnier-Orr diagrams, Jackson diagrams, action diagrams, entity life cycles, state-dependency diagrams and various matrices like the create, read, update and delete (CRUD) matrix.	• Process logic description techniques are represented by action diagrams under OO techniques because action diagrams encompass all the other diagrams plus concurrency not covered by any of them.
10. **Linguistic techniques:** Language action perspective (LAP).	• The overview of language action perspective (LAP) in Dietz (2003).

Table 6-2: Modelling techniques to be studied

6.4 Design science research

6.4.1 Background

Design science research provides another view complementing the positivist and interpretive perspectives for IS research. It distinguishes between natural science and the science of the artificial, and concentrates on phenomena that are created (designed artefacts) rather than objects occurring naturally. Designed artefacts can be, among other things, algorithms, HCI constructs, ISD methodologies and ISD techniques (Hevner et al., 2004).

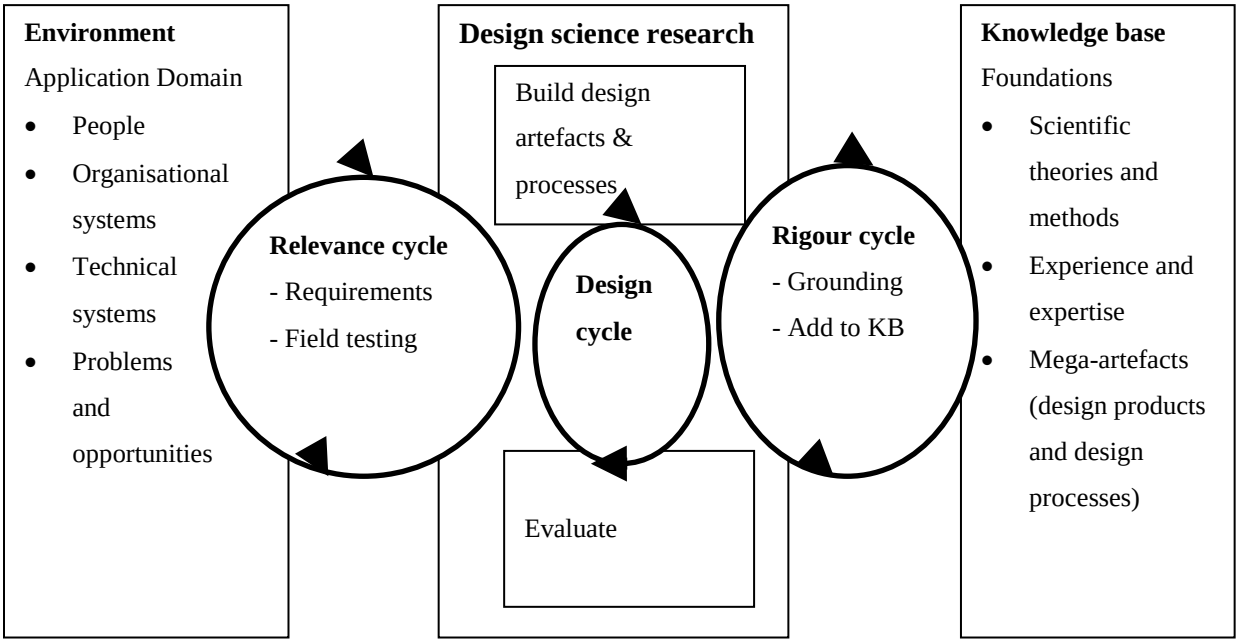
Iivari (2007) postulates an epistemology of design science consisting of the following three types of knowledge:

- Conceptual knowledge contains no truth value and describes concepts, constructs, classifications, taxonomies, typologies and conceptual frameworks.
- Descriptive knowledge contains truth value and describes observational facts, empirical regularities, theories and hypotheses. For instance, *X causes A in situation B.*
- Prescriptive knowledge contains no truth value and describes design product and design process (technological rules and norms) knowledge. For instance, *in order to achieve A, do* act_1 *...* act_n and *if you want A and you are in situation B, you should do X*.

Hevner (2007) considers design science research to be consisting of three areas: the **design science research** itself, the contextual **environment** of the research and the existing **knowledge base** informing the research, as well as three closely related activity cycles influencing these three areas (see Figure 6-3):

- The **relevance cycle** takes requirements form the environment of the research and places them in the research domain. It also takes artefacts created during the research and places them in the environment for field testing.

- The **rigour cycle** provides grounding theories, methods, domain experience and expertise from the foundations knowledge base to the research. It also adds new knowledge to the knowledge base generated from the research.
- The **design cycle** supports the research activity for the creation and evaluation of design artefacts and processes.



Hevner (2007)

Figure 6-3: Design science research cycles

6.4.2 Research methodology

According to Vaishnavi and Kuechler (2004), design science research involves the following steps: an *awareness of a problem* stated in a proposal, a *suggestion* defined in a tentative design, the *development* of an artefact, an *evaluation* of the artefact with performance measures and a *conclusion* with results. Peffers et al. (2008) and Geerts (2011) elaborate on that and identify six activities (see Table 6-3).

DSRM activities	Activity description	Knowledge base
Problem identification and motivation	*What is the problem?* Define the research problem and justify the value of a solution.	Understand the problem's relevance and its current solutions and their weaknesses.

DSRM activities	Activity description	Knowledge base
Define the objectives of a solution	*How should the problem be solved?* In addition to general objectives such as feasibility and performance, what are the specific criteria that a solution for the problem defined in step 1 should meet?	Knowledge of what is possible and what is feasible. Knowledge of methods, technologies and theories that can help with defining the objectives.
Design and development	*Create an artefact that solves the problem.* Create constructs, models, methods or instantiations in which a research contribution is embedded.	Application of methods, technologies and theories to create an artefact that solves the problem.
Demonstration	*Demonstrate the use of the artefact.* Prove that the artefact works by solving one or more instances of the problem.	Knowledge of how to use the artefact to solve the problem.
Evaluation	*How well does the artefact work?* Observe and measure how well the artefact supports a solution to the problem by comparing the objectives with observed results.	Knowledge of relevant metrics and evaluation techniques.
Communication	Communicate the problem, its solution, and the utility, novelty, and effectiveness of the solution to researchers and other relevant audiences.	Knowledge of the disciplinary culture.

Geerts (2011)

Table 6-3: Design science research methodology

These six activities were approached as follows in this study:

Step 1: Identify problem and motivate: There is a non-trivial gap between business modelling and ISD modelling (see section 1.3). ISD modelling is precise and has many widely used modelling techniques to enable it. Business modelling is mostly done using textual descriptions without a specific technique employed.

Step 2: Define the objectives of a solution: The solution to this problem is to define an integrative modelling technique that will bridge the gap between business and ISD modelling. The objectives of this technique are specifically as follows:

- **Ease of use:** This technique must be simple enough so that non-technical users must be able to use it with minimum training.
- **Expressiveness**: At the same time, the technique must be expressive enough so that detailed ISD level modelling can be derived from it without further interactions with users or analysts.

This can be stated in a different way. Luukkonen, Korpela and Mykkanen (2010) developed a classification of modelling techniques used in the earlier phases of the ISD life cycle. They identified two dimensions: degree of structure related to the syntactic aspect of model (ranging from unstructured to highly structured), and scope related to the semantic and pragmatic aspects of the model (ranging from technical to human and organisational). In terms of this classification, the purpose of the integrative modelling is to be highly structured and also to cover the full range on the scope dimension.

Step 3: Design and develop: In Chapter 7, such an integrative modelling technique is developed using a grounded approach and qualitatively analysing existing modelling techniques.

Step 4: Demonstrate: In section 8.2, this technique is demonstrated by looking at a number of very common business modelling situations where the output will be used by ISD modelling techniques. Various case studies will be used to illustrate the application of the proposed modelling technique. The case studies will incorporate organisational, business and IT aspects and also the different phases of a systems development life cycle.

Step 5: Evaluate: In section 8.2, this technique is evaluated by comparing it with a number of existing integrative modelling techniques while considering the objectives of the research identified in step 2 above.

Step 6: Communicate: The purpose of the thesis.

6.4.3 Research output

Vaishnavi and Kuechler (2004) developed a taxonomy of design science research output (see Table 6-4 below).

	Output	Description
1	Constructs	The conceptual vocabulary of a domain
2	Models	A set of propositions or statements expressing relationships between constructs
3	Methods	A set of steps used to perform a task – how-to knowledge
4	Instantiations	The operationalisation of constructs, models and methods
5	Better theories	Artefact construction as analogous to experimental natural science

Vaishnavi and Kuechler (2004)

Table 6-4: The outputs of design science research

According to Iivari (2007), design science research should be based on a sound ontology. He bases his view on Popper's three worlds: World 1: material nature, World 2: consciousness and mental states, and World 3: institutions, theories and human artefacts. He contends that Hevner et al.'s (2004) classification of IT artefacts (constructs, models, methods and instantiations) is too general and cannot easily be applied. He states that design science should be based on a proper typology of IT artefacts answering the question of what is designed and built in IS to distinguish it from computer science and software engineering. According to him, IS is primarily interested in IT applications and he therefore provides seven archetypes of IT applications based on function and role (see Table 6-5).

Role/function	Metaphors	Examples
To automate	Processor	Many embedded systems Many transaction processing systems
To augment	Tool (proper)	Many personal productivity systems, CAD
To mediate	Medium	Email, instant messaging, chat rooms, blogs, Electronic storage systems (CDs and DVDs)
To inform	Information source	Information systems proper
To entertain	Game	Computer games
To provide art	Piece of art	Computer art
To accompany	Pet	Digital (virtual and robotic) pets

Iivari (2007)

Table 6-5: Archetypes of IT applications

Venable (2006) asks a similar question on the kind of knowledge design science research should produce. He firstly mentions the outputs as given in Table 6-4 and then states that it should also produce clear and complete guidelines and advice for practitioners in selecting the most appropriate solution/technology and implementing it, as well as create knowledge statements with the purpose of verification and improvement by other researchers.

The artefact of this study is an integrative modelling technique between business and ISD. In terms of the design science research outputs as defined in Table 6-4, this modelling technique can be seen to incorporate the first three outputs: constructs, a model and (to some extent) a method. This technique must be evaluated to determine if it is an improvement on existing techniques. This evaluation must be done according to the specified criteria of **simplicity** and **adequate expressiveness** as defined in step 2 in the previous section.

6.4.4 Research guidelines

Hevner et al. (2004) provide seven guidelines for design science research:

1. The research must produce a feasible design **artefact.**
2. The purpose of the research is to develop solutions to **relevant** and important business **problems** using technology-based solutions.
3. The design artefact must be evaluated using well-executed **evaluation** methods to show the utility, quality and efficacy of the artefact.
4. The research must show clear and demonstrable **contributions.** Contributions include the artefacts themselves as well as new foundations and new methodologies.
5. The research must show **rigour** in both the development and the evaluation of the artefact.
6. Design is a **search process** for an effective artefact, while considering other competing approaches.

7. The results of the research must be **communicated** to both academic (satisfying rigour requirements) and management-oriented (satisfying relevance requirements) audiences.

Venable (2010) warns, however, that there is a lack of consensus on the relative importance of the guidelines, demonstrated by a survey done among IS scholars who write, review, edit and publish design science research articles. Although the guidelines in general are approved it is advised not to use them less mechanistically than is currently done. Many respondents warned against using the Hughes et al. (2000) guidelines as a mandatory checklist for evaluating the research. Venable (2010) suggests a cumulative model rather than a subtractive model (inherent in the checklist approach) for evaluating research.

6.4.5 Design Science Research Theory

According to Venable (2006), design science research in the IS field has mostly excluded the creation and testing of theory and left it to natural and social sciences. He contends that theory should be a primary output of any research, including design science research. For him, theory is the distinguishing factor between design science and design practice.

Gregor (2006) created a taxonomy of theory types in IS research. She distinguishes five types as follows:

I. **Analysis.** *Says what is*. It fundamentally involves the analysis and description of phenomena without considering their causal relationships. An example of such a theory is the dynamic framework for classifying ISD approaches and methodologies (Iivari et al., 2001a).

II. **Explanation.** *Says what it is, how, why, when and where*. It provides an explanation without trying to predict precisely and has no testable propositions. An example of such a theory is the structurational model of technology by Orlikowski (1992).

III. **Prediction.** *Says what is and what will be*. It provides predictions and has testable predictions, but cannot justify causal relationships. Examples of such a theory are very rare in IS and a related theory is Moore's law of doubling processing power every 18 months while cost stays the same.

IV. **Explanation and prediction (EP).** *Says what is, how, why, when, where and what will be*. It provides predictions, has testable propositions and causal explanations. An example of such a theory is the theory that shows the causative drivers and emergent mechanisms driving temporal changes in user beliefs and attitude towards IT usage (Bhattacherjee and Premkumar, 2004).

V. **Design and action.** *Says how to do something*. It gives explicit prescriptions for building an artefact. An example of such a theory is the design theory for systems that support emergent knowledge processes (Markus, Majchrzak and Gasser, 2002).

Gregor (2006) specifies the structural components common to all theory as follows:

- **Means of representation,** for instance, in words, mathematical formula, diagrams, pictures and even prototypes.
- **Constructs,** which are the phenomena of interest and includes observational terms, theoretical terms and collective terms.
- **Statements of relationship** between constructs, for instance, associative, compositional, conditional and causal.
- **Scope**, indicating the level of generality of the relationship statements (indicated by indicators such as "some", "many", "all" and "never") as well as statements of boundaries limiting generalisations.

She further specifies the structural components contingent on theory purpose as follows:

- **Causal explanations** are statements explaining the causal relationships between phenomena.
- **Testable propositions (hypotheses)** are empirically testable statements of relationships between phenomena.

- **Prescriptive statements** are theoretical statements that specify how practical goals can be achieved.

Venable (2006) discusses the important distinction between design science research and design practice. In summary, it can be said that design practice use knowledge or apply technology to a particular, situated problem, while design science produce knowledge or invent technology for a generalised class of problems. This knowledge is communicated to academic and industry stakeholders via design theories. Walls, Widmeyer and El Sawy (1992:40–41) provide the following seven characteristics that distinguish design theories:

1. Goals are intrinsic to design theories (to achieve goal X, do Y), while they are extrinsic to explanatory and predictive theories (Y causes X).
2. A design theory can never involve pure explanation or prediction.
3. Design theories are prescriptive.
4. Design theories are composed of kernel theories from existing knowledge.
5. Explanatory theories tell *what is*, predictive theories tell *what will be*, normative theories tell *what should be*, and design theories provide the *how to/because*.
6. Design theories show how other theories can be put to practical use.
7. Design theories are theories of procedural rationality.

6.5 The journey

In this section, the journey that was taken during the research is explained. The reason for this is to show why the research is constructed the way it is. In essence, the research itself stayed the same, but the methodology changed halfway through the study as a result of the progression the research took.

6.5.1 Part 1 – Grounded approach

When the decision was made to use grounded theory for data analysis, the typical mistake was made of underestimating what it was all about. During the first attempt certain problems surfaced. Firstly, it was difficult to determine what a code was.

Much of the literature on grounded theory does not really explain or illustrate what codes really are. It became imperative to read more on grounded theory and to especially look at examples of coding.

The second problem was handling the sheer volume of data. Because of this problem, it was decided to use some sort of software support to help manage the data. In the first attempt, no specific computer-assisted qualitative data analysis software (CAQDAS) was used. Initially, only Microsoft Word was used to record findings. Table 6-6 shows an example of how concepts were represented during the first attempt, while Table 6-7 shows how codes were added to the same table at a later stage (please note: at this stage a column to show how these techniques were represented was also included, but this was discarded later on).

The third problem, related to the amount of data, is not including referencing information right from the beginning. At some stage during analysis, the need arises to link back to the original source material. Because this was only done very informally, in an effort to facilitate the referencing of codes, the open coding data was done using Excel and referencing columns added (see Table 6-8). Another form of referencing is also needed (as illustrated in Table 6-9) to show how codes developed out of initial concepts.

A fourth problem during the first attempt was using the grounded approach as a sequence of fairly distinct phases, i.e. the idea was to first do open coding, then axial coding, then selective coding, and so on. The result of this was the generation of a massive amount of codes without any context. It was soon realised from personal experience and by re-reading Glaser (2003) that the different phases of the grounded approach run concurrently. It implies that while one is looking for concepts and codes, one already has to start with axial and selective coding.

No	Concept	Representation	Notes
123.	Entity	Box with different parts	The things of interest to an organisation or system about which we want to store information
124.	Entity type vs. entity instance/occurrence	N/a	
125.	Attribute	Text within Entity	A descriptive value associated with an entity. It is a property of an entity
126.	Relationship	Lines	An association between two entities. There may be more than relationship between the same two entities. A relationship normally arises because of association (e.g. customer places order) or structure (order consists of order-lines)
127.	Cardinality	Symbols (e.g. n, m), numbers or arrow heads	The number of entities related to another entity. Cardinality can be one-to-one, one-to-many and many-to-many.
128.	Primary key	Double underlined attribute, double asterix,	The attribute/s that uniquely identifies an entity instance.

Table 6-6: Example of recording concepts during the first attempt

No	Concept	Representation	Code	Notes
1.	Entity - The things of interest to an organisation or system about which we want to store information	Box with different parts	Thing – informational (semiotic)	Every physical (e.g. car, product), informational (e.g. invoice) and conceptual (e.g. strategy, transaction) thing; as well as human "things" (e.g. client, department) are represented by an entity. The code "thing" was chosen because other possible codes for this concept like "object" and "entity have very specific meanings in the field of ISD. Therefore, the more generic term "thing" was chosen.
2.	Entity type vs. entity instance/occurrence	N/a	Relationship – Type to instance	
3.	Attribute - A descriptive value associated with an entity. It is a property of an entity	Text within entity box	Thing attribute	An attribute can be seen purely as a property of the entity, but can also be seen as a free standing object with a structural relationship to the entity, e.g. an employee entity consists of
4.	Relationship - An association between two entities.	Lines with text	Relationship	There may be more than one relationship between the same two entities.
5.	Relationship type	N/a	Relationship type	Class diagrams are more explicit in distinguishing and defining different kinds of relationships. A relationship arises because of: • Association (e.g. customer places order)

Table 6-7: Example of the development of codes during the first attempt

Area	Source	Section	Page	No	Quote	Concept	Property	Value
OO	Pender (2002)	Part II Session 5 Understanding the Use Case Model	49	1	They called the process a Use Case driven approach because it views the system from the perspective of how external entities (people and other systems) need to interact with the system.	Model	Element	External entity
OO	Pender (2002)	Part II Session 5 Understanding the Use Case Model	49	2	The called the process a Use Case driven approach because it views the system from the perspective of how external entities (people and other systems) need to interact with the system.	External entity	Type	People System
OO	Pender (2002)	Part II Session 5 Understanding the Use Case Model	49	3	The Use Case model is a collection of diagrams and text that together document how users expect to interact with the system.	Modeling	Medium	Diagram Text
OO	Pender (2002)	Part II Session 5 Understanding the Use Case Model	49	4	The Use Case model is a collection of diagrams and text that together document how users expect to interact with the system.	Modeling	Element	User interaction with system
OO	Pender (2002)	Part II Session 5 Understanding the Use Case Model	49	5	The Use Case model focuses on the critical success factors of the system, in terms of the functionality or features that the users need to interact with.	System	Element	Features Functionality

Table 6-8: Example of open coding using Excel highlighting referencing of code sources

Action	Transaction (SQL), transaction ACID properties (SQL), Activities (IDEF0), Input (IDEF0), Output (IDEF0), Units of behaviour (IDEF3), Transition (IDEF3), Data flow (DFD), Process (DFD), Activity (RAD), Operation (ALC/OLC), behaviour, operation (OO), scenario (UML-UCD), Use case (UML-UCD), typical course of events (UML-UCN), alternate course (UML-UCN), interaction (UML-sequence), activities (UML-activity), input (UML-activity), output (UML-activity),steps (UML-activity), actions (UML-activity), flows (UML-activity), state transitions (UML-state chart), Transformation (SSM-Root), activities (SSM-Conceptual), how (Zachman), action (decision tree/table), statement (structured English),

Table 6-9: Example of linking a code ("action") back to its original concepts

At a certain stage during the first attempt there was a general feeling of being lost in the data. The whole exercise has become almost quantitative instead of qualitative. The approach became mechanistic and yielded no new insights into the data. The conclusion was that the use of a proper CAQDAS tool would solve the problem. It was decided to get ATLAS.ti to handle the management of all data.

Although using the software tool alleviated some of the problems, it indirectly caused some other problems. The main problem that was solved with ATLAS.ti was the issue of referencing source information. Figure 6-4 shows how codes are created directly linked to the original source document. Figure 6-5 illustrates how it is possible to display all references together with the original quotations for a specific code.

One of the biggest problems with using a tool like ATLAS.ti is that it is now possible to easily create even more codes than was possible before. When the codes are initially created and one would like to start doing further axial coding, the available codes are presented in alphabetical order as illustrated in Figure 6-6 (there were 56 codes in total in this one source document in the example). It became very difficult to work from that point onwards because the codes have lost their “context”. One has to perform a very time-consuming comparison of codes, often going back to the original codes to determine what the code was all about. (The memo facility is very helpful with this.) The end result of the initial axial coding is shown in Figure 6-7. Again, there was the problem of information overload. It was found that it was easier to break the overall network into subnetworks, as shown in Figure 6-8.

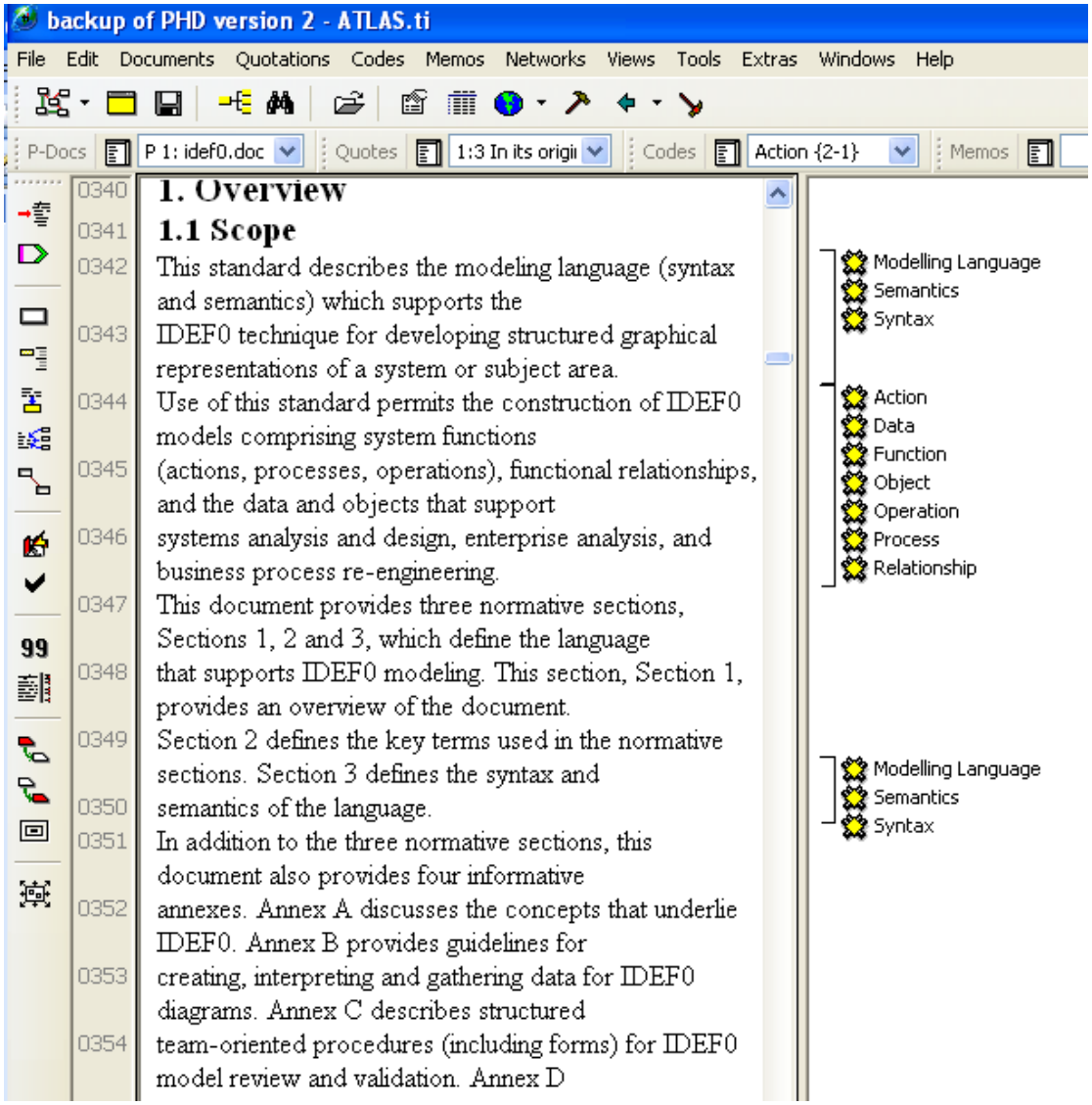


Figure 6-4: Example of creating codes in ATLAS.ti

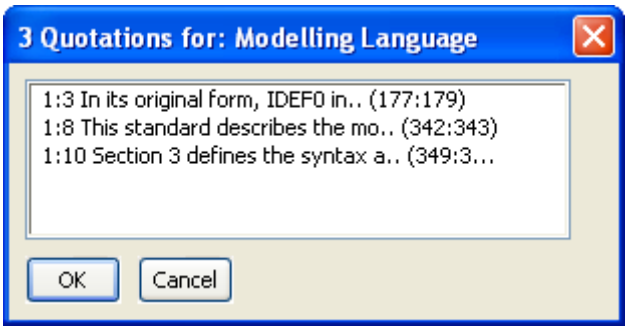


Figure 6-5: Linking a code ("modelling language") with its original quotations

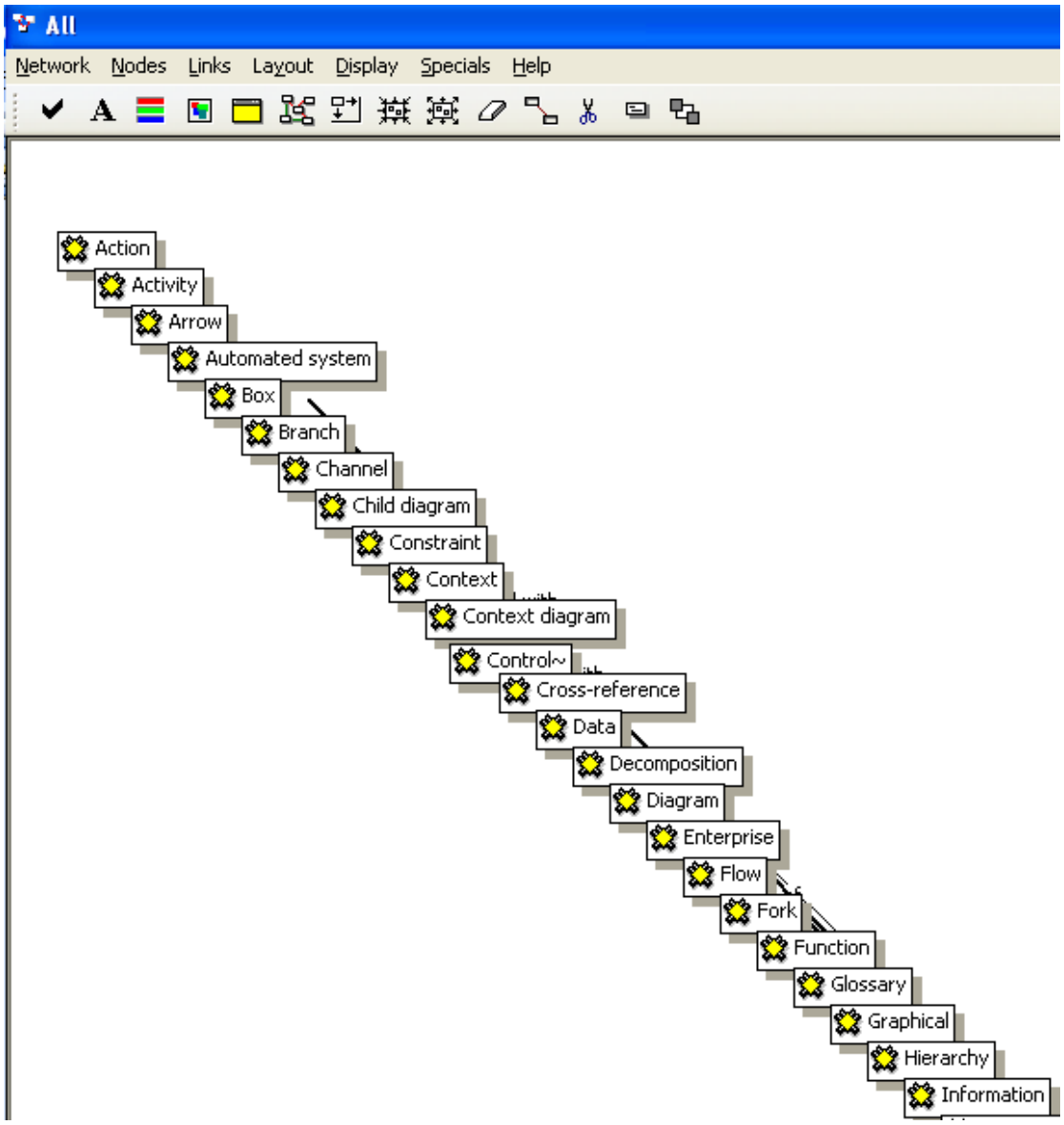


Figure 6-6: A list of codes before axial coding

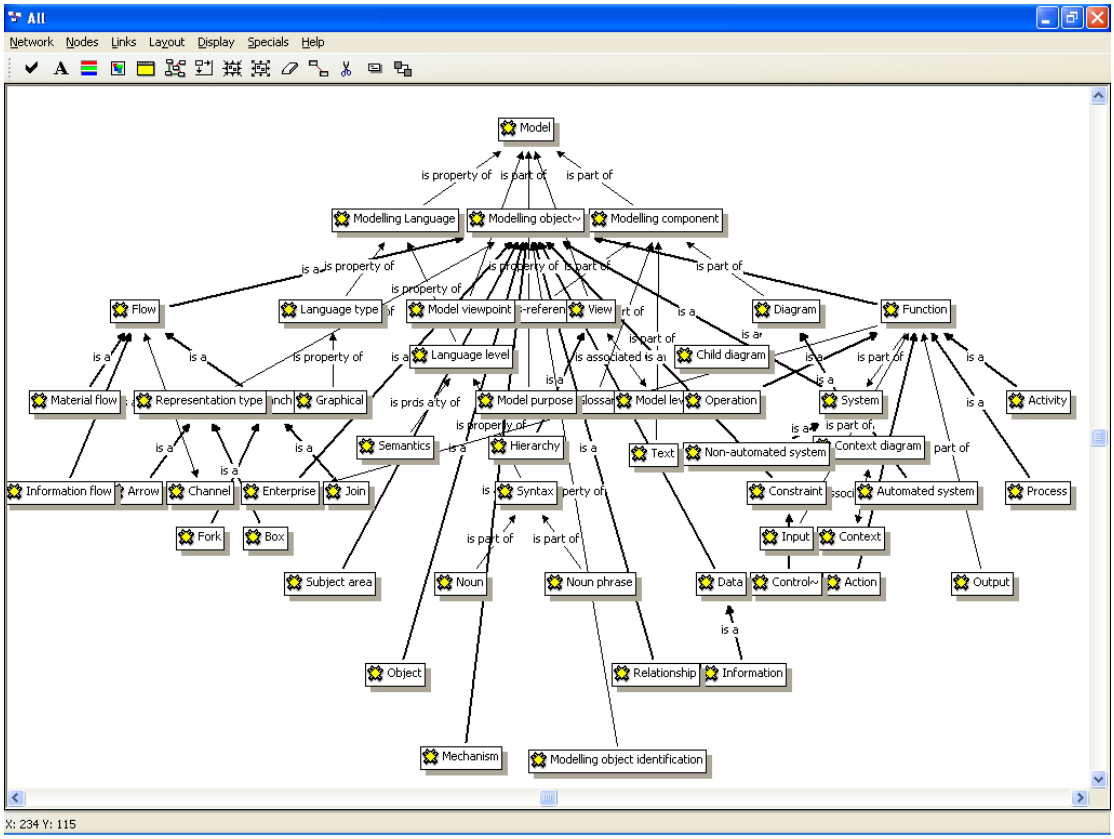


Figure 6-7: A comprehensive network of codes in ATLAS.ti

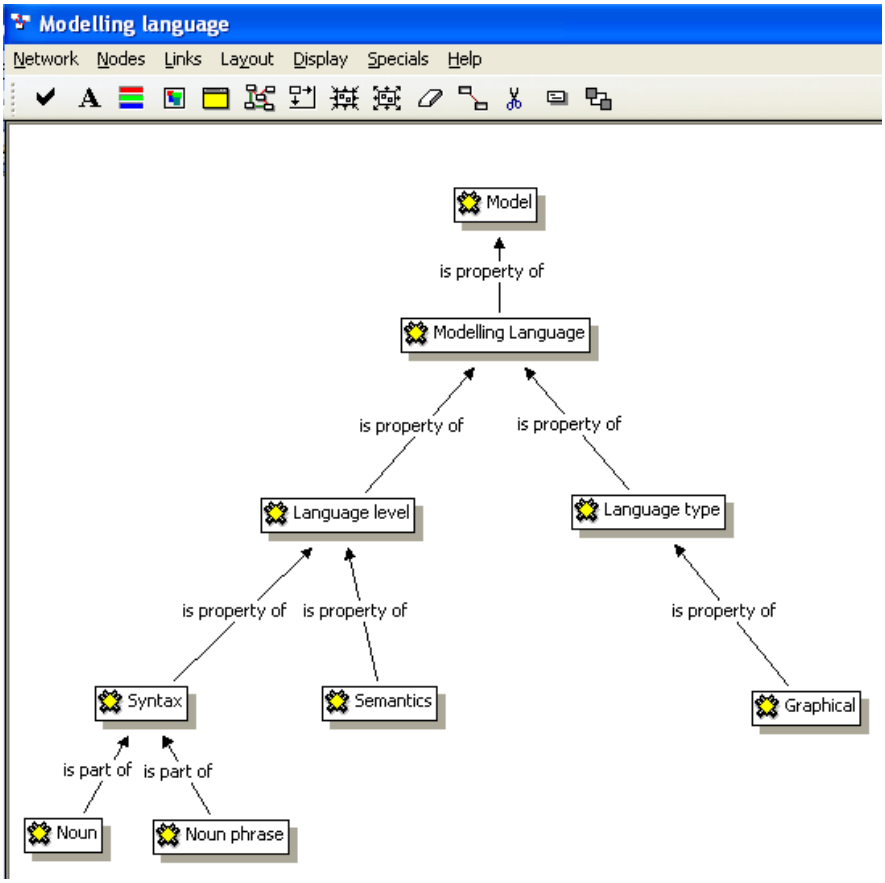


Figure 6-8: A specific network of codes in ATLAS.ti

6.5.2 Part 2 – Design science research

At a certain stage in the process, it became clear that this study was not really a traditionally grounded theory study. The subject matter studied is not social behaviour, but rather conceptual design artefacts. The grounded approach has helped tremendously to complete the first part of the study, namely creating an integrative business and ISD modelling technique. However, to complete the study, a design research approach was followed to evaluate and improve the artefact created by the first part of the study.

6.6 Conclusion

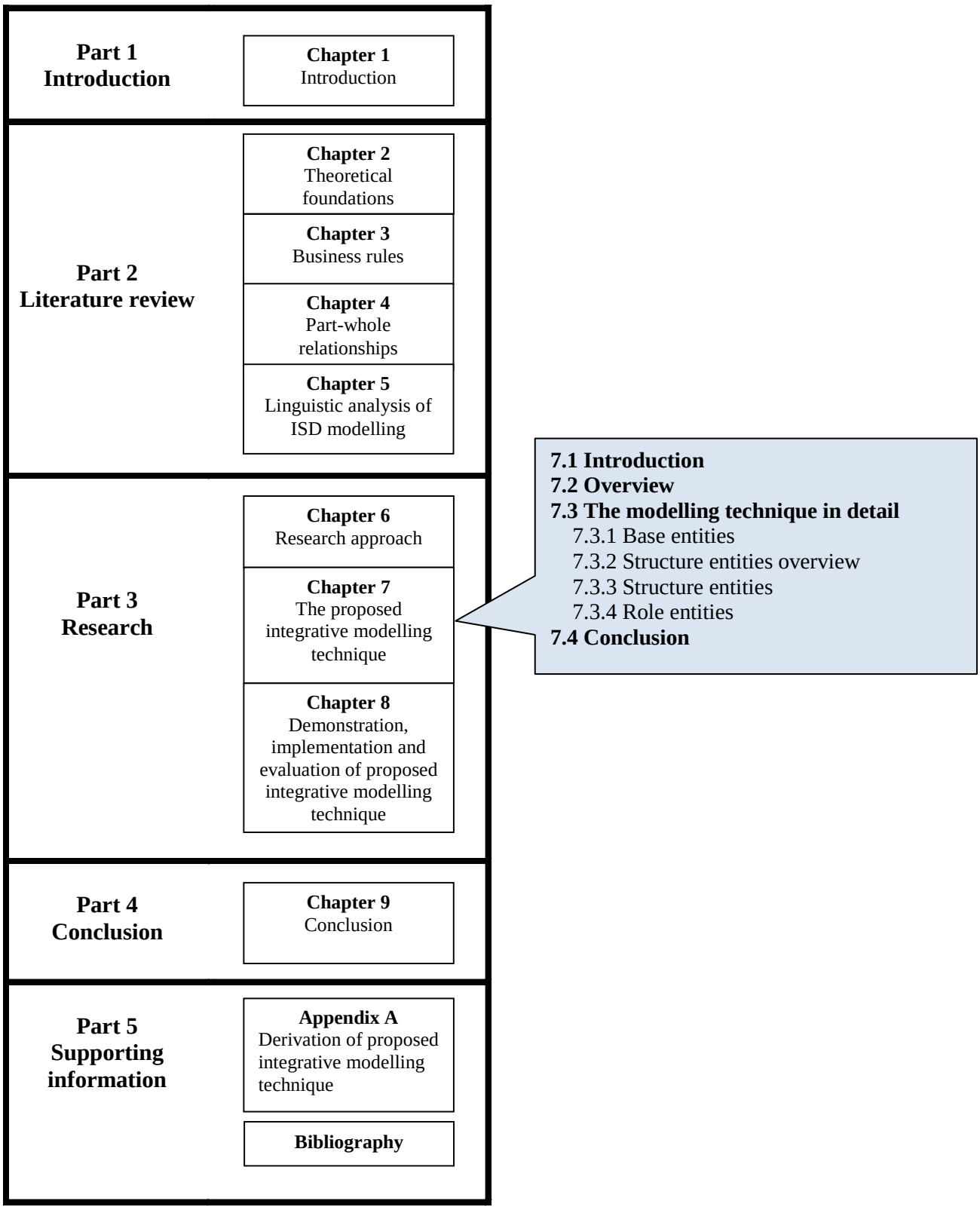
The theory types and structural components of theory as specified by Gregor (2006) can be used to classify the contribution that this study attempts to make (see Table 6-10).

Theory types	I. Analysis, II. Explanation and V. Design and action.
Means of representation	Mostly visual diagrams with textual explanation.
Primary constructs	Business and ISD modelling constructs such as *actor*, *object*, *action* and *event*.
Statements of relationship	• The relationships between the primary constructs, such as *actors initiate actions* and *actors can either be individuals or organisations*. • The levels of primary constructs, such as the *base level* corresponding to morphology in linguistics, *structure level* corresponding to syntactics in linguistics and *role level* corresponding to semantics in linguistics. • The relationships of these primary constructs to existing modelling techniques, for instance, an *actor* in the theory corresponds to an *actor* in use cases and *external agent* in DFDs.
Scope	Business modelling with the purpose of informing ISD modelling.
Causal explanations	Not present.
Testable propositions	Not present.
Prescriptive statements	Explicitly, mostly in the method description, for instance, *only use phrases in active voice and not passive voice*. Implicitly, by limiting phrases to only contain a certain fixed set of possible values.

Table 6-10 Theory types and components of proposed integrative technique

Although the search is on for fundamental real mechanisms and structures out there (ontologically), this study can make no claim to have found all of these mechanisms and structures quantitatively (all of them) or qualitatively (exactly right). Only a subset of techniques was used. Although a wide and representative subset was chosen, it is possible that a *real* exist for which there is currently no *empirical* or *actual* evidence.

7. The Proposed Integrative Modelling Technique



7.1 Introduction

The research done in this thesis consists of two parts: The first part is a qualitative, grounded approach study of a number of modelling techniques. The proposed integrative technique described in this chapter is the end result of that study. More detail on how this technique was derived from the underlying data is explained in more detail in Appendix A. The second part of the study is a design science research analysis and evaluation of this proposed integrative modelling technique and is discussed in more detail in Chapter 8.

7.2 Overview of the modelling technique

One of the main findings of this study is that modelling is closely linked to linguistics (Chapter 5). The research process and subsequent literature study on linguistics clearly showed that a linguistic-type framework is the best way to structure the integrative technique in a consistent, coherent and integrated way. Therefore, the fundamental **modelling entities** of business and ISD can be divided into three categories (see Figure 7-1):

- **Base entities:** corresponding to the morphological level in linguistics
- **Structure entities:** corresponding to the syntactical level in linguistics
- **Role entities:** corresponding to the semantic level in linguistics

The **base entities** are the most basic building blocks in the modelling process. They represent the real-world objects that make up organisations and systems. In the same way that verbs, nouns, adjectives, adverbs, pronouns and others make up the words of natural languages, these entities form the **words** of the proposed modelling language.

The **structure entities** form, like their natural language counterparts, the **model sentences** and **phrases** with which systems, organisations and situations can be described. They use the base entities plus specific language constructions to form these model sentences and phrases. A number of model sentences together form a **model**. In turn, specific subsets of the model form **model views**.

The **role entities** provide insight into the meanings of the base and structure entities. For example, an actor can either play an agent role or patient role in a model phrase.

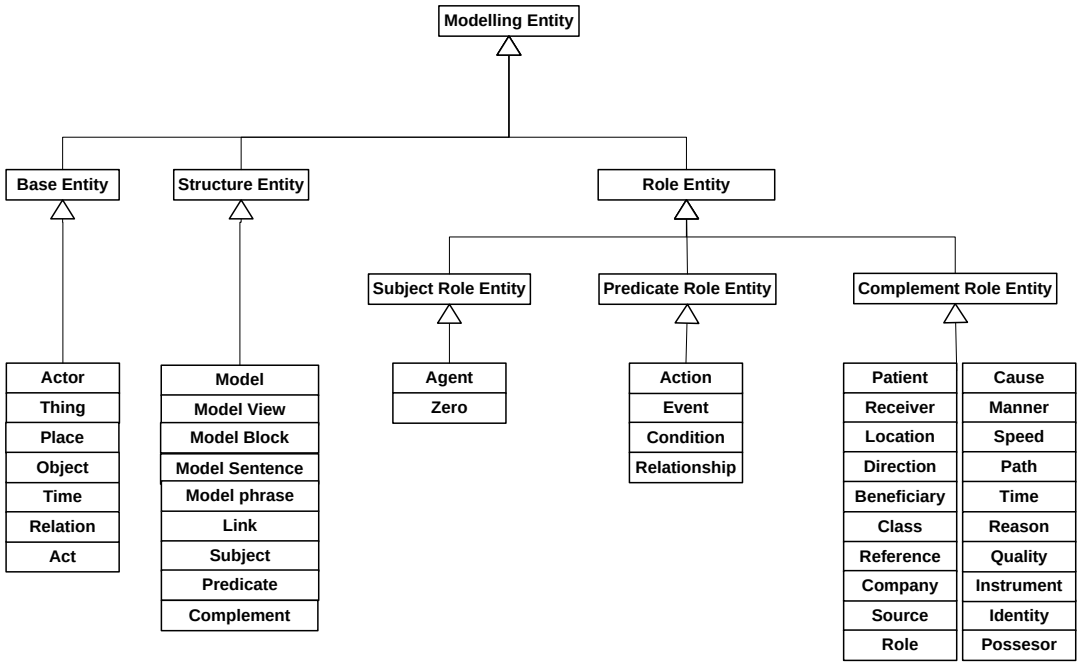


Figure 7-1: A high-level overview of the modelling entities

A note on the choice of modelling entity names is important at this stage. In the history of ISD modelling, various names have been used for the same modelling entities. It makes it very difficult to decide on a name that has so many synonyms. The approach followed in this study was to use names that are either the most popular names (used by the most techniques), or to use names that are more general rather than more specific (for instance, rather *thing* than *entity* or *object*), or to use novel names when many different, conflicting names were used.

7.3 The modelling technique in detail

Each of the modelling entities introduced in the previous section are described in more detail in this section. The discussion is divided into three parts:

- In section 7.3.1, **base entities** are discussed in more detail as follows:
  - Base entities overview
  - Actors (intelligent things)
  - Objects (non-intelligent things)

  - Acts and relations
- In section 7.3.2, **structure entities** are discussed in more detail as follows:
  - Structure entities overview
  - Models and model views
  - Model blocks
  - Model sentences and phrases

- In section 7.3.3, **role entities** are discussed in more detail as follows:
  - Role entities overview
  - Actions
  - Relationships
  - Conditions
  - Role entity analysis (showing how model phrases and sentences can be analysed to determine their role entity types)

7.3.1 Base entities

7.3.1.1 Base entities overview

From the research, it became clear that to model any information system or organisation, surprisingly few entities are needed. Corresponding to the morphological level of linguistics, these base entities are as follows (see Figure 7-2):

- **Things:** intelligent entities (actors) and non-intelligent entities (objects) corresponding to nouns and noun phrases in linguistics (section 5.2.1.3 (a)).
- **Actors** (intelligent things): human, institutional and artificial entities that can act and make decisions. Actors correspond to animate nouns in linguistics (section 5.2.1.3 (a)).
  - **Human actors** (or persons) are individual humans like clients, employees and users.
  - **Institutional actors** (or human activity systems) are groups of people where the group has an own identity and is involved in purposeful activities, for

example, companies, departments and project teams. Institutional agents are made up of other base entities.

- **Artificial actors** are human-engineered artefacts that simulate human behaviour, such as bank account systems, pocket calculators and robotic manufacturing systems. Artificial actors consist of other base entities.

- **Objects** (non-intelligent things): physical, conceptual or informational (semiotic) entities employed during actions by agents. Objects can consist of other objects. Objects correspond to inanimate nouns in linguistics (section 5.2.1.3 (a)).
  - **Physical objects** are material things that can be detected by the senses, such as raw material, products, furniture and vehicles.
  - **Conceptual objects** (or **information)** are concepts that are created by humans to make their world more understandable and to communicate with other actors. Three important specific types of conceptual objects are listed below:
  - **Places (*or* locative conceptual objects)** are physical two- or three-dimensional areas or conceptually demarcated areas. Examples include countries, geographical areas, residential plots, offices in a building and areas on a screen, form or report.
  - **Times (*or* time-related conceptual objects)** are indications of absolute (e.g. 12/01/2011) or relative (e.g. two days after month-end) times.
  - **Model blocks** are structure modelling entities (described in the next section) that can be handled as if they are base entities. For instance, the process *withdraw money from ATM* (a model block) can be seen as a single conceptual object and used as such, although it consists of a number of base entities and their relations and acts.
  - **Informational objects** are combinations of information and physical objects (acting as the media) created with the specific purpose of capturing, storing and displaying information, like files, databases, books, spreadsheets, input forms and whiteboards. Informational objects are related to conceptual objects in that they both have information as basis, but the physical medium of a conceptual object is not of importance.
- **Acts and relations** are dynamic relationships (acts) and static relationships (relations) between things corresponding to verb and verb phrases in linguistics.

Relations are common to all situations and comprises a (theoretically) finite specific set of auxiliary verbs and verb phrases, such as *is-a-type-of*, *consists-of*, *has*, *is-above*, and *is-equal-to*. Acts, on the other hand, are specific to every situation and consists of all active verbs that can be used in a situation under discussion, for example *withdraw* (money), *print* (receipt), *deposit* (money), *transfer* (money) and *pay* (beneficiaries) in the ATM situation. Acts and relations correspond to verbs in linguistics (section 5.2.1.3 (b)).

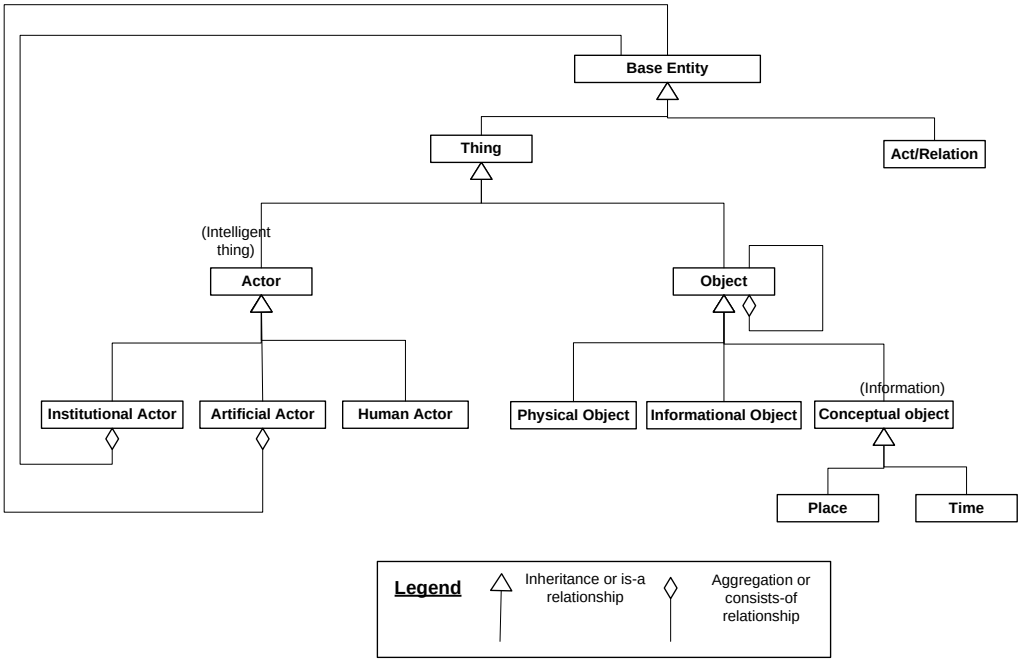


Figure 7-2: Modelling base entities

To illustrate base entities, the modelling case study of an ATM system is used. The base entities of a typical ATM system are listed in Table 7-1 below.

Base entity type	**Entity**
Human actors	• Client • ATM operator • Branch teller
Institutional actors	• Bank • Other banks • Branch
Artificial actors	• Bank system • ATM system
Physical objects	• ATM machine • ATM keyboard

Base entity type	Entity
	• ATM screen • ATM envelope receiver • ATM printer • ATM card reader • ATM money dispenser • Envelope • Paper (for printer)
Informational objects	• ATM card • Transaction receipt • Cash
Conceptual objects (information)	• Account • Transaction • PIN
- Place	• ATM location
- Time	• End of day
- Model block	• Withdraw money from ATM process
Acts	• Withdraw (money) • Deposit (money) • Print (bank statement)
Relations	• Consists-of • Is-type-of

Table 7-1: ATM example – base entities

7.3.1.2 Actors (intelligent things)

(a) Overview

An actor refers to any entity that autonomously performs actions and makes decisions. It is mostly human beings, but includes institutional actors (like organisations, branches and departments) and artificial actors that behave like humans (for instance, robots, computer programs and ATM banking systems) (see Figure 7-3).

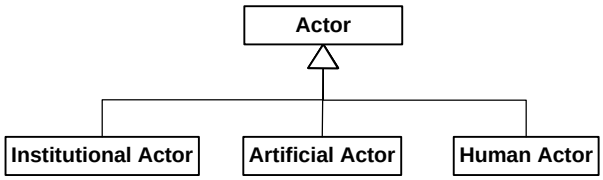


Figure 7-3: Types of actors

When considering actors, there will seldom be an interest in a specific instance of an actor (client 2345), but rather in the kind of actor (client), although specific instances could be of interest, especially if there is only one instance per type in a specific context (e.g. the Reserve Bank of South Africa, Nelson Mandela or Einstein). Flowing from this is the concept that when we consider actors, we consider the **roles** they play in an organisation or system. Roles are either linked to acts (e.g. teacher, manager, service provider) or to relations with other entities (e.g. owner, shareholder, employee and wife).

Human actors are persons, individuals or humans like patients, clients, employees and users. From a business perspective, the following issues are of interest when considering human actors: people are seen as a means of production; human resources are seen as a factor of production; end consumers have needs (Maslow's hierarchy of physiological, security, social, self-esteem and self-actualisation needs), and business stakeholders are shareholders, owners, employees, consumers, competitors, intermediaries and suppliers.

An **institutional actor** is a group of humans and organisations that has an own, separate identity from the humans and organisations making it up, for example, a family, a company, a department, and a project. An institutional actor can also be seen as a human activity system created for purposeful activities (Checkland, 2000).

From a business perspective, institutional actors have to do with enterprise types such as private, government, semi-government and non-profit institutions, enterprise forms like sole proprietors, partnerships, companies (private and public), closed corporations, cooperatives and trusts, all of the various functions in an organisation like purchasing and marketing (a function like purchasing has an organisation, the purchasing department, to execute it), organisational culture and management style.
An institutional actor can, for instance, be a project team that is formed to create a specific artefact such as a building and consist of human actors as team members (e.g. builders and architects), physical things like tools and raw material, locations like the building plot and the project office, actions like build, draw, plan, approve, get loan, and paint, and events like builder holidays, material running out and project deadlines.

An institutional actor can also be a family that consists of human actors like parents and children, physical things like furniture, cutlery, clothes and a car, locations like the family house and family members' rooms, actions like eat, make food, change nappies, mow the lawn, and wash the dishes, and events like baby feeding times, children's birthdays, various licence renewal times and grocery shopping.

Artificial actors are human-engineered artefacts, with an own identity, that simulate human behaviour, for instance, bank cheque systems, hand calculator systems and robotic manufacturing systems. Typically, an artificial actor is owned and managed by a legal entity that is responsible for its actions. For instance, in an automated contract negotiation situation or in an automated purchase decision, a legal entity may be represented by an artificial actor.

Artificial actors can either be manipulating information or manipulating physical entities, or both. Examples of artificial actors manipulating data are IS and programs, while artificial actors manipulating physical entities are things like robotic systems, automated mail sorting systems and numerical control machines.

In a theoretical sense, there is no difference between an institutional actor and an artificial actor. They can both consist of exactly the same types of entities. The only difference is the extent to which humans are involved in the system. In the case of an institutional actor, humans and groups of humans are the main actors, while in the case of an artificial actor, computers are the main actors with humans playing a supportive role.

(b) Components

A distinction can be made between primitive/atomic and composite entities. Primitive/atomic entities cannot be divided any further into other entities, while composite entities can be divided into other entities. The modelling entity actor can be divided into composite parts, as shown in Figure 7-4.

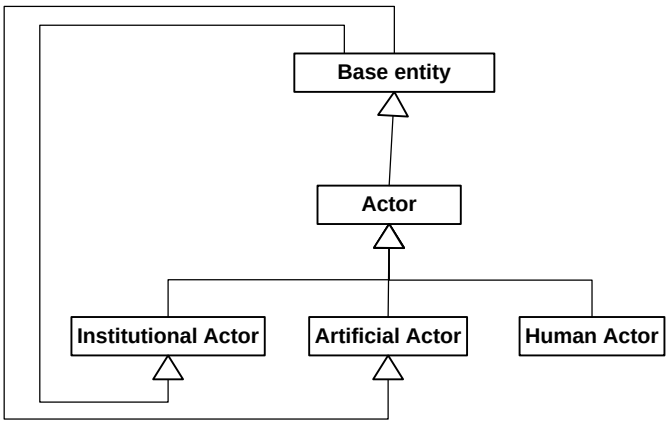


Figure 7-4: The composite relationships of actors

For the purpose of modelling organisations and IS, a **human actor** is normally seen as atomic/primitive and cannot be divided into components any further. A doctor or psychologist will also consider the biological, psychological and mental parts of a human agent and could, therefore, divide humans into further parts.

Institutional actors, on the other hand, are always composite entities consisting of many other materials, autonomous and conceptual entities. For instance, a bank consists of other *institutional actors* like divisions and departments, *human actors* like managers, tellers and secretaries, *physical objects* like furniture, buildings and computers, *informational objects* like accounts, statements, bank cards and contracts, *places* like plots of land and offices in buildings, and *acts* like creating a new account, withdrawing money and printing a statement.

Artificial actors are also, like institutional actors, composite entities consisting of other entities. Simpler artificial actors like traffic robots consist of *physical objects* like bulbs, coloured lenses, poles, cables and *artificial actors* like a CPU and a timer. More complex artificial actors, such as a cellphone system, consist of *institutional actors* like maintenance departments, *human actors* like technicians and developers, *physical objects* like transmission towers, cellphones and SIM cards, *informational objects* like accounts, messages, phone calls, *places* like computer rooms, and *actions* like making a call, sending a message and renewing a contract.

(c) Categories

The type of category into which actors can be placed depends on the roles they play:

- **Initiator:** starting/triggering an action
- **Source:** providing input for an action or agent
- **Actor**/doer: doing something in an action
- **Owner:** owning an agent, action, thing or place
- **Decider:** making decisions in an action
- **Manager:** managing an agent, thing, location or complex action (involving other actors or objects)
- **Receiver:** receiving output from an action or actor
- **Client**/beneficiary: receiving value from an actor, action, object or place

Actors can also be categorised according to the creation of other actors, actions, objects or locations. These can be seen as meta-roles. Some examples are the following:

- **Analyst** of a business
- **Designer** of a system or **architect** of a building
- **Developer** of a system or **builder** of a bridge

Every **institutional actor** has a number of stakeholders (actors/roles) involved with it. It includes all the roles in the organisational structure as well as the roles of people and groups who are not part of the organisation. These are as follows:

- **Customers** or beneficiaries of the products and services of the organisation.
- Internal and external **service providers** (e.g. credit bureaus, information providers and the Legal Department).
- The **organisation** itself. Any people or groups that are part of the organisation. This is normally indicated by means of the organisational structure.

- **Regulators** are any organisations that have a controlling or regulating influence on the system under discussion (e.g. the government tax services could require a monthly report on cars leaving the country).

The categories into which **artificial actors** can be placed are (following the classification in the AOR technique) as follows:

- Software agents, e.g. involved in e-commerce transactions.
- Embedded systems, e.g. in automatic teller machines.
- Robots, e.g. in manufacturing.

(d) Properties

Some of the main properties of actors are the following:

- Some form of **unique identification** – normally a name (or names) and one or more numbers, e.g. internationally a passport number, nationally a country identification number or company registration number, in a company a staff number, system number or department number, and in other organisations a member number.
- **Responsibility** for other actors, things or locations, or for performing certain actions. Responsibility is given when certain transactions, such as a legal transaction (e.g. getting married, entering into a contract, buying a property, registering as a company), institutional transaction (e.g. appointment as employee, promotion to manager, being given access rights, installing a system), and social transaction (e.g. becoming a father, becoming a member of a society, being appointed captain of the team, sponsoring an event) occur. Actions can be seen as the way by which actors discharge their responsibilities.

(e) Actions

Two main types of actions that can be performed on actors are meta- (or life cycle) actions and operational actions. Every actor has a life cycle and **meta-actions** are related to these life cycle actions. For instance: a client gets registered, has transactions with the organisation and gets deregistered; an employee gets appointed,

works for the organisation and either leaves, retires or dies; a company gets registered, operates and gets deregistered or is taken over by another company; and an information system gets analysed, designed, developed, tested, implemented, goes operational and eventually gets decommissioned. **Operational actions** are the actions performed on actors on a day-to-day basis.

The following actions can be done on (in contrast to actions done by) human actors[3]:

- **Transform** or process *human actor1* into *human actor2*. For example, process private citizen to become a soldier/prisoner, promote employee to a manager, give beauty treatment to client and possibly do a heart transplant on patient.
- **Transport** or distribute *human actors* from *location1* to *location2*. For example, drive passengers to their destination.
- **Store** or house *human actors* in *location*. For example, guests stay in the hotel and the soldiers live in the barracks.
- **Exchange** or trade *human actors* for *things (of value).* In the modern world this mostly means human actors exchanging their time for money, but in the past and in some places in the world today it can also refer to slavery. For example, contractors work for a fee, employees work for a salary and soccer player gets transferred to new club for a specific amount.
- **Control** or regulate *human actors* from *state1* to *state2*. For example, ensure all new employees have valid identification documents (change status to "ID validated"), arrest citizens breaking the law (change status to "arrested"), count people in a census (state change from "not counted" to "counted") or authorise an employee to a specific action (state change from "unauthorised" to "authorised").

Using the classification of the LAP technique (Dietz, 2003), there are three kinds of actions that can be done to actors. They are the following:

- Material and immaterial **production actions** (transform, transport, store, exchange and control), as described by the previous paragraph.

[3] Here and everywhere else in this chapter where different kinds of actions are discussed the Magee and de Weck (2004) taxonomy discussed in section 2.6 is used as basis.

- **Coordination actions** between *human actor1* and *human actor2*. This entails entering into commitments (becoming responsible) to perform production actions and complying with the commitments. A commitment by one party creates a claim by another party on the outcome of the commitment. For example, client takes out a home loan and commits to repay monthly instalments. The bank then has a claim on the monthly payments and if payments are not submitted, the bank has a claim on the home itself.
- **Communication actions** are used to perform coordination acts. Communication acts can be divided into the following:
  - **Formative actions** between *human actor1* and *human actor2*. It involves establishing and maintaining a communication channel between the actors. It involves expressing, transmitting and receiving information without any distortions (physical, empirical or syntactic). For example, "Can you speak louder, I can't hear you", "I have received your email and will study it" or "The diagram is illegible, can you resend in a different format".
  - **Informative actions** between *human actor1* and *human actor2*. It involves the establishment of an intellectual understanding of the coordination action between the participants. It involves informing *actor2* and confirmation by *actor2* of the facts without distortion (semantic and pragmatic). For example, "I understand the conditions of the contract, except point 2.7, can you please explain"; and message on screen: "By pressing the button you agree to pay the fees as indicated".
  - **Performative actions** between *human actor1* and *human actor2*. It involves the actors committing to perform the production actions required by the coordination action. For example, "I do take this man as my lawful husband", or pressing the "Accept" button and signing a contract.

The following actions can be done on institutional actors (using the Magee and De Weck (2004) taxonomy discussed in section 2.6):

- **Transform** or process *institutional actor1* into *institutional actor2*. For example, changing a private company to a public company and an engaged couple getting married and becoming a family.

- **Transport** or distribute *institutional actors* from *location1* to *location2*. Not really applicable. Components of an institutional agent can be transported, but usually not the institutional actor as a whole (except perhaps families emigrating).
- **Store** or house *institutional actors* in *location*. For example, the family lives in a house, the United Nations head office is situated in New York.
- **Exchange** or trade *institutional actors* for *things (of value).* For example, selling shares in company, taking over a company for a cash amount.
- **Control** or regulate *institutional actors* from *state1* to *state2*. For example, making an organisation COBIT-compliant (i.e. change state from non-COBIT- to COBIT-compliant), performing a yearly audit on a company (change state from unaudited to audited for that year).

The following actions can be done on artificial agents:

- **Transform** or process *artificial actor1* into *artificial actor2*. For example, assembling an information system from web services.
- **Transport** or distribute *artificial actors* from *location1* to *location2*. For example, delivering a cellphone to a client, installing an information system at a branch.
- **Store** or house *artificial actors* in *location*. For example, the information system is situated in the computer room.
- **Exchange** or trade *artificial actors* for *things (of value).* For example, buy or rent PC and software.
- **Control** or regulate *artificial actors* from *state1* to *state2*. For example, test information system (change state from “untested” to “tested”).

7.3.1.3 Objects (non-intelligent things)

(a) Overview

Objects are non-autonomous, non-intelligent entities and can either be physical, informational or conceptual. A **physical object** is any natural entity or human-created artefact that has a material, time-space aspect to it. It includes anything tangible, like products, raw material and tools. An **informational object** is the resultant entity

created to store and communicate information, for example, a manual file, a computer file, a database, a book, a data capture input form, a whiteboard, a label on a product and a movie. An informational object always consists of a medium (a physical object) and information (a conceptual object). A **conceptual object** (or information) originates in the mind of an actor. If the actor needs to communicate the concept to another, he needs to code it (e.g. text, mathematical symbols, pictures, sound or video) onto a medium (e.g. paper, stone, signboard, magnetic strip, tape, or even human body as in sign language). Specific types of conceptual objects are **time** and **place**.

Physical objects can serve as follows:

- They can be used to achieve a specific goal like **tools**.
- They can be used as input into a process like **raw material**.
- They can be created, i.e. they are the actual **products**.

In business terms, physical objects relate to the means of production, like equipment, raw material and factors of production such as natural resources and assets.

In computer terms, physical objects relate to hardware like desktops, notebooks, tablets, servers, mainframes, PDAs and cellphones; hardware components like CPUs, memory, I/O interfaces and storage; peripherals like input devices and output devices; and networks like LANs, wireless LANs and WANs.

Although animals could be seen as a separate base entity, for normal business purposes animals are not treated as actors but rather like physical objects that will always be the objects acted on by other actors.

Any piece of information can be translated to any type of **informational object.** For instance, the same client information can be captured via an input form, entered into a client file, displayed on a computer screen, printed on a report, archived to microfiche.

Informational objects can consist of physical entities like paper and ink; as well as other informational artefacts like a book consisting of pages or a brochure consisting

of pamphlets and a CD. Informational objects have a specific medium like paper, magnetic and electronic. The duration of informational objects differs and has an effect on the type of medium used. It can be temporary like RAM, blackboard notes, permanent like hard drives, CDs, DVDs, s, or archived.

Conceptual objects or information is meaningful, organised data that exist for a specific purpose, for instance, client information, a list of clients, the contents of a movie, a framework like TOGAF, the score of a symphony, the lines and colours of a drawing, and the movements of a dance.

A place is a special type of conceptual object and is any spatial area like a plot in a city, an office in a building, a province in a country, a position on screen, an area on a report. Note that a place is always in relationship to something else. A place is indicated either by an address or a coordinate in a two- or three-dimensional coordinate system or some relative position indicator like *the top row*. Place address can both be physical (e.g. at longitude x and latitude y) or conceptual (e.g. the third item of an array).

Depending on the situation, the places can be specified by using representations like maps, office layouts and network diagrams. Examples are a map of Europe or the world with the rental branches and service depots indicated on it or a list of branches and service depots per country and per town, the layout of any current networks that have a bearing on the system under discussion, and any place that is not physical, e.g. client/server, back office/front office and web.

Objects can be places or contain places. For instance, some informational objects can be seen as places, such as a database or a data warehouse at a specific address (location). Physical objects like filing cabinets are to some extent places and contain subplaces.

In business terms, place is related to land and buildings as a means of production, to primary location factors like potential market, infrastructure, raw material and labour, and to secondary place factors like climate, government intervention, political situation, business premises, capital and personal considerations.

A place can consist of other places, for instance, a plot can be divided into subplots, a building can be divided into floors and offices and a web screen can be divided into frames.

(b) Components

Objects are made up of other objects. In summary, physical objects can either be atomic (e.g. a screw) or made up of other physical objects and/or informational objects (e.g. a car consisting of parts and an instruction manual); conceptual objects can consist of zero or more conceptual objects (e.g. a coding system consists of codes); and informational objects consist of zero or more informational objects as well as zero or more physical objects (e.g. a book consists of chapters, paper and glue) (e.g. Figure 7-5).

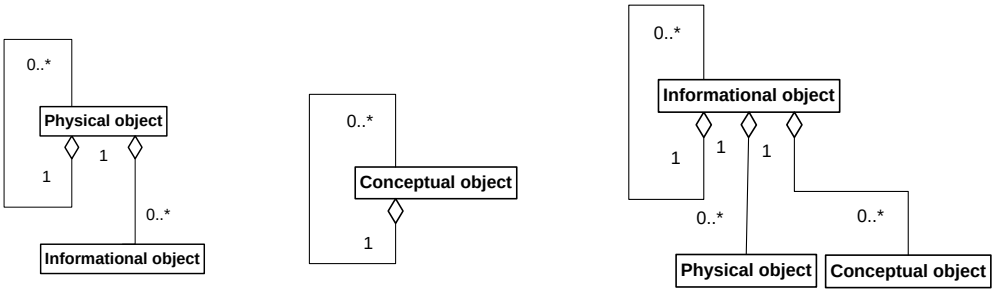


Figure 7-5: The components of objects

Primitive/atomic physical objects refer to physical objects that cannot be divided any further. This includes product components like screws, glass panes and a window frame, and undividable raw material[4] like a specific length of steel pipe.

Composite physical objects are made up of other physical objects, e.g. a product made up of components (which are physical objects). On a bigger scale, these hierarchies of physical objects are called a Bill of Material (BOM). Composite

[4] Raw materials, of which the unit of measure is not *each*, are problematic when considering dividing it further. When one uses, for instance, steel pipe as raw material, one needs a certain length of it. Theoretically, one can divide it infinitely into smaller lengths, but for the purposes of the product, the correct measure (in whatever unit of measure) is the smallest possible division of the raw material and therefore a primitive/atomic thing.

physical objects can also be made up of physical objects as well as informational objects, e.g. a product with a barcode, label and/or price tag.
Combining a physical object, like a product, with conceptual objects, like the label information, results in a physical entity when the intent of the conceptual objects is purely to be a component of the product.

When a conceptual object is combined with any physical media for the purpose of manipulating or storing information, the end result is an informational object. The same conceptual object combined with different physical media results in different informational objects, for instance, client information can be entered on a client registration form or a computer screen, printed on a paper report or displayed on a computer screen, and stored on CD, DVD, hard drive or even a paper-based file.

The combination of primitive atomic-level conceptual objects (e.g. specific client's name plus specific client's surname) produces a higher-level conceptual object (e.g. a specific client's full name).

The different categories of conceptual objects are divided as follows:

- **Operational information,** normally in tabular format (seen as a spreadsheet: horizontally, a column like client name, and vertically, a row representing a specific client) in files, databases or spreadsheets can be divided as follows:
  - **Groups** of data like databases, file systems and spreadsheets.
  - **Entities** like tables, files and worksheets.
  - **Columns** like fields and attributes.
  - **Tuples** like rows and records.
  - **Cells**[5] representing actual values.
- **Published information** in text format, like books, manuals and brochures, is typically structured as follows:
  - Books
  - Chapters

[5] Theoretically, information can be divided even further into characters and even bits and bytes.

  - Sections
  - Paragraphs
  - Sentences
  - Words
  - Characters
- When information is **web-based,** the same principles apply, but the divisions are different, e.g. pages, frames, etc. Multimedia also extends this format.
- **Audio-based information,** like music albums (as on CD), is divided into tracks.
- **Video-based information** (as on DVDs) is divided into arbitrary segments (e.g. index, scene 1, and director's comments). On a more technical level, a video can be divided into frames.

Every "tuple" or row of data will always have some fields that can be used to identify the tuple. These fields are called **keys.** Many different fields are potential keys and these are called *candidate* keys. *Composite* keys consist of more than one field. The main, unique identifying key is called the *primary* key. If a primary key is referenced in another entity, it is called a *foreign* key.

Every field has a specific data *type,* like text, numbers, images, Boolean and date. Every field has a *domain* – a set of possible values. A field can also have a *range* – a minimum and maximum value within which values can fall. The format of a field defines the structure of the field, for instance, a date can be in the formats yyyy/mm/dd or mm/dd/yy, and the format of a phone number is "(area code) 999 9999".

(c) Categories

The most common categories of **physical objects** are the following:

- **Raw material:** typically some object unprocessed from nature that is either mined (like iron, sand, salt and crude oil), or harvested (like grain, rice, meat, fish and fruit)
- **Products for everyday usage**, like perfume, food products, clothes, stationery, medicine and car oil

- **Tools**, like hammers, drills, surgical knives and tin openers
- **Machines**, like bulldozers, food processors, microwave ovens and personal computers

The most common categories into which **conceptual objects** can be placed are the following:

- **Textual** information like words and characters
- **Audio** information like speech, sounds and music
- **Visual** information like images

The most common categories of **informational objects** are the following:

- **Data storage** entities, like databases and files for storing information.
- **Publishing** entities, like books, magazines and brochures.
- **Multimedia** entities, like photos, song tracks, video clips and movies.

Three categories of places can be identified:

- **Physical places**: geographical and two- or three-dimensional areas.
- **Conceptual places**: e.g. backoffice/frontoffice, client/server.
- **Networks** have both physical and conceptual elements, e.g. IP address and physical node.

(d) Properties

All objects have the following:

- **State**. All objects have one or more states that change as the result of actions or events occurring. For instance, a physical or informational product can have a life cycle state that goes from “being manufactured” to “manufactured” to “ordered” to “delivered” to “available” to “written off” to “sold” or “destroyed”. If the product is rented (rental car or library book), the rental state can go from “available” to “rented” to “late rental” to “missing”.

Some of the additional properties of **physical objects** are as the following:

- **Unit of measure**. Physical objects when used as raw material or as a product has some unit of measure, like “each”, “meter”, “metre squared”, “kilogram”, “litre”, “dozen”, ream. This includes the dimensions and weight of the object.
- **Type of material**. The constituent material, such as sand, paper, steel, aluminium, wood, satin, bamboo.
- **Other properties** like colour.

Some of the main properties of **places** are the following:

- **Address**. Every place must have some sort of address to allow access to the place. The address can be physical or conceptual (like an IP address).
- **Entity held**. Every place stores or holds specific entities (or a combination of entities), for instance, a house stores *human actors* and their *physical objects*; a library stores books and other *informational objects*; a network node stores *artificial actors* like a network operating system; and the screen holds screen controls.

(e) Actions

The following actions can be done on physical objects (using the Magee and De Weck (2004) taxonomy discussed in section 2.6):

- **Transform** or process *physical objects* + *informational objects* (optional) into *physical objects*. Combining a physical object, like a product, with informational objects, like a label, results in a physical entity when the intent with the informational object is purely to be a component of the product. For example, creating products from components and raw materials, making food from ingredients and building houses from supplies.
- **Transport** or distribute *physical objects* from *Location 1* to *Location 2*. For example, transporting raw material from the mine to the factory, driving a car

from home to work, pumping oil from the ship to the refinery and delivering orders to clients.

- **Store** or house *physical objects* in *location*. For example, storing stock in warehouse, parking cars in the garage.
- **Exchange** or trade *physical object* for *value object*. For example, selling products for monetary compensation.
- **Control** or regulate *physical object* from *State 1* to *State 2*. For example, doing quality assurance of product, stress-testing a new car and destroying expired food.

The following actions can be done on informational objects (many of these actions also involve a physical medium):

- **Transform** or process *Informational object 1* into *Informational object 2*. For example, compiling a report from various pieces of information, printing a report (i.e. change the medium from electronic/magnetic into paper).
- **Transport** or distribute *informational object* from *Location 1* to *Location 2*. For example, sending email to a client, couriering a book to student, SMS text message to friend, send video clip.
- **Store** or house *informational object* in *location*. For example, storing a spreadsheet on a computer (at an address), storing books in library at specific place (using the Dewey system as an addressing system).
- **Exchange** or trade *informational object* for *object (of value).* For example, company selling their music videos, lending a library book.
- **Control** or regulate *informational object* from *State 1* to *State 2*. For example, reviewing a book before publishing.

The *transform* actions that can be done on conceptual objects are the following:

- Meta-actions:
  - **Define,** e.g. plan a book, design a database (create SQL script), design a cellphone contacts list.
  - **Create,** e.g. publish book, create database (run SQL script), create cellphone contacts list. All of these involve using a physical medium as well.

  - **Change,** e.g. publish new version of book, change the structure of the database and change the structure of the cellphone contacts list.
  - **Destroy,** e.g. destroy book, drop (permanently delete) database, remove the cellphone contacts list from the cellphone.
  - **Rename** (a specific type of change), e.g. change the name of the book, change the name of the database, rename the cellphone contacts list from "Contacts" to "My contacts".
- Data manipulation actions (mostly known as CRUD actions):
  - **Create** (insert in SQL) information, e.g. place a new paragraph in a book, insert a new record into a database table and add a new contact to the cellphone contacts list.
  - **Read** (select in SQL), e.g. read, copy or copy a page from the book, read one or more records in the table, search for one or more contacts in the cellphone contacts list.
  - **Update,** e.g. replace one or more words, sentences, sections, chapters of the book, change one or more fields of a record in a table, change any of the details of a contact in the cellphone contacts list.
  - **Delete,** e.g. remove one or more words, sentences, sections or chapters of a book, delete one or more records in a table and remove a contact from the cellphone contacts list.
- Control actions:
  - **Grant/revoke** access rights to actor, e.g. give the key to the cupboard where the book manuscript is stored to the editor, give full update rights to the accounts system, allow one's secretary to view one's contacts on one's cellphone.

The following actions can be done on places (using the Magee and De Weck (2004) taxonomy discussed in section 2.6):

- **Transform** or process *place* into *place*. For example, divide the office into suboffices with dry-walling, subdividing a screen into segments.

- **Transport** or distribute *place* from *Place 1* to *Place 2*. Only really possible to transport an object containing places between places, e.g. move file cabinet to the next office.
- **Store** or house *place* in *place*. Only really possible to store an object containing places in a place, e.g. store file cabinet in office.
- **Exchange** or trade *place* for *thing (of value)*. For example, sell a house, rent a mailbox.
- **Control** or regulate *place* from *State 1* to *State 2*. For example, transfer property deed, inspect offices and regulate access to IP addresses.

7.3.1.4 Act/relation

The base entities of "act" and "relation" correspond to those of verbs in morphology. An **act** relates to an active verb in an action (for example, the man *moves* the car). A **relation** relates to verbs and auxiliary verbs in an event (for example, when stock *becomes less than* reorder level), a relationship (for example, the house *consists of* rooms) and a condition (for example, if employee *is a* manager).

These base entities only really make sense when they are used in model phrases and sentences (structure entities) and play specific predicate roles (role entities) in these phrases and sentences. Therefore, acts and relations will be discussed later when these concepts are discussed.

7.3.2 Structure entities

In natural language, words can be structured together to form sentences. In a similar way, base entities can be structured together with specific language constructs to form **structure entities** (see Figure 7-6). Structure entities correspond to the syntactical level in linguistics (section 5.2.2).

7.3.2.1 Models and model views

(a) Overview

A distinction must be made between the “real” and the “representational” (or modelled) world. Business and ISD modelling involves taking a part of real-world organisation, representing it using some technique like UML or ERD to represent the real system *as-is*, and then designing a new real system *to-be*. This representation of the new system to-be is then developed and implemented to become part of (or change) the real system. If a new system must be developed on the same part of the real-world organisation, the created information system is now part of it.

Although there is one real world, there can be many views (representations) of it. A view will always only represent a part of the real world and will also represent only a specific aspect of it. For instance, in an ERD, all other aspects are ignored (although they are as important) and the focus is only on the data aspect of the system.

The model represents all information available on the real-world situation, organisation or system modelled. By implication, the model (it does not matter how complete or detailed) is only a representation of the real world. From this one model, various **model views** can be produced, representing any specific aspect that the model viewer wants to emphasise.

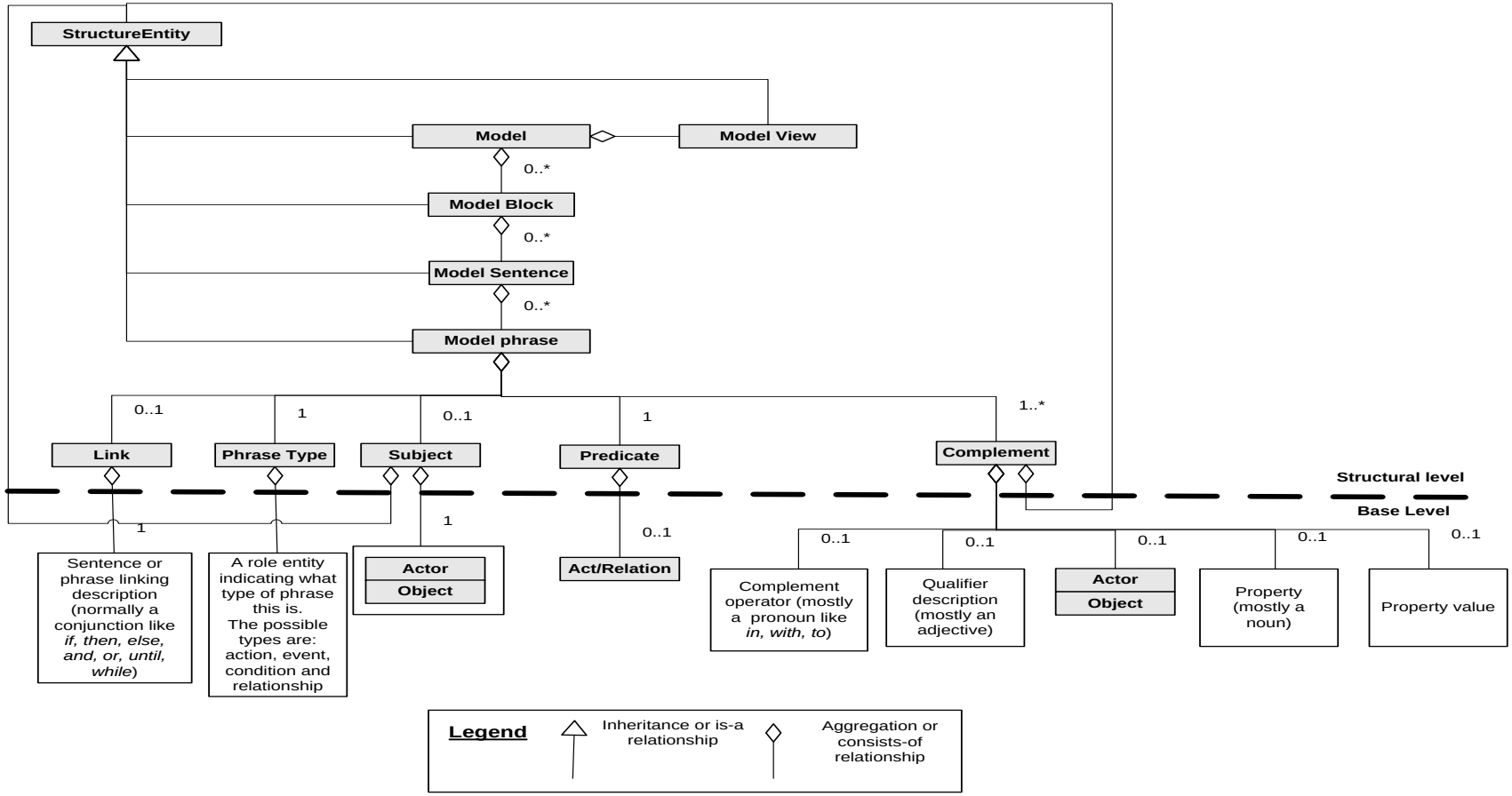


Figure 7-6: Modelling structure entities

(b) Categories

Views can be based on a number of aspects. The first aspect is the type of **base entity** that needs to be considered. It can be seen as a horizontal view. For instance, data modelling concentrates on *informational objects*, an organogram on *human* and *institutional actors*, a work breakdown structure on *actions* and a bill of material on *physical objects*.

The second aspect is the **level of detail** that needs to be expressed. It can be seen as a vertical view. For instance, using use case modelling, sometimes only the systems and subsystems involved are considered (model block level), other times only case names are used (action level) and at other times, the focus is on the use case detail such as steps, actors, conditions and business rules (action step level).

The third aspect is **stakeholder** perspectives. It can combine characteristics of the first two aspects. The rows *Planner* (scope, contextual), *Owner* (enterprise), *Designer* (system, conceptual, logical), *Builder* (physical, technical) and *Subcontractor* (component) of the Zachman framework (section 2.3) illustrate the typical stakeholder perspectives well. Over time (during the life cycle of the system), different perspectives will be more important than other. There is also the design issue of postponing certain decisions until a later phase. For instance, during conceptual design, concentrate on how the system is to operate but without taking the specific technology or implementation into account.

A fourth aspect is the **history** view. It can be combined with any or even all of the previous views. An as-is view shows how things are currently while a to-be view shows how things are designed or planned to be in the future. In practice there is not just one as-is and one to-be view. Different versions of both these views can exist.

A fifth aspect is that of **linguistics**. In this view, the entities and their relationships are seen as the elements of a language. As in any language, the various linguistic levels of morphology, syntax, semantics and pragmatics are considered.

7.3.2.2 Model blocks

Model phrases and model sentences can be grouped into **model blocks.** They are named for reference and usage. Once a model block has been created, it becomes a conceptual object and can be used as such. For instance, it is possible to create a model block called "ATM login" representing a use case or business process and consisting of a number of model sentences and phrases:

Link	Subject	Predicate	Operator	Qualification	Complement
When	Client	Inserts			ATM card
			Into		ATM card reader
If	ATM card	Is-not		Valid	ATM card
Then	ATM system	Displays			Error message "Invalid card"
	ATM system	Returns			ATM card
			From		ATM card reader
Else	ATM system	Displays			Request PIN screen
If	PIN	Is-not-equal-to		Stored	PIN
Then	ATM system	Displays			Error message "Invalid PIN"
	ATM system	Returns			ATM card
			From		ATM card reader
Else	ATM system	Displays			Transaction options screen

A model block has the following three properties, over and above the constituent model phrases and sentences:

- A **name** by which it can be referenced and used either as a conceptual object or as an executable set of instructions.
- Zero or more **input parameters** that can be used to let the model function like a subroutine or function when it is used as an executable.
- Zero or more **output parameters** generated from the execution of the model block.

A created model block can be used like any other conceptual object, for instance:

Subject	Predicate	Operator	Qualification	Complement
Analyst (*human actor*)	Maintains (*act*)			ATM login (*conceptual object – model block*)
Client (*human actor*)	Initiates (*act*)			ATM login (*conceptual object – model block*)
ATM login (*conceptual object – model block*)	Is-a-type-of (*relation*)			Use case (*conceptual object*)

Or it can be used as an executable, for instance:

Subject	**Predicate**	**Operator**	**Qualification**	**Complement**
Chef (*human actor*)	Executes (*act*)			Pancake recipe (*conceptual object – model block*)
		Using		[Number of people] (*input parameter*)
ATM system (*artificial actor*)	Executes (*act*)			ATM login (*conceptual object – model block*)
Financial program (*artificial actor*)	Executes (*act*)			Future value (*conceptual object – model block*)
		Using		[Rate] (*input parameter*)
		Using		[Number of payment periods] (*input parameter*)
		Using		[Payment per period] (*input parameter*)
		Returning		[Future value] (*output parameter*)

7.3.2.3 Model sentences and phrases

Model phrases are joined together to form **model sentences**. Model phrases and sentences are either events, conditions, actions or relationships.

The most fundamental structure entity is the **model phrase**. In its simplest format, it consists of the basic parts of the simplest sentence in natural language, namely subject-predicate-object. They can be read from left to right as a complete humanly understandable sentence. For instance, the following model phrases describe both actions and relationships between the base objects of the ATM system.

Subject	**Predicate**	**Object**	**Role type**
Client (*human actor*)	Withdraws (*act*)	Cash (*informational object*)	Action
Client (*human actor*)	Has (*relation*)	Accounts (*conceptual object*)	Relationship

The two model phrases above can be read and understood by humans, for instance “client withdraws cash” and “client has accounts”. Conversely, these model phrases can be created from natural language statements. But, importantly, the structured approach in creating model phrases make them also open to manual or system-assisted analysis and manipulation. For instance, it is possible to algorithmically

create a partial-use case diagram for the first phrase and a partial non-attributed ERD for the second. It is also possible to lay down specific rules for these phrases. For instance, the subject of an act can only be an actor, while the subject of a relation can be an actor or an object.

This model phrase structure can be extended to also indicate other objects involved in an action or relationship, over and above the direct object. The **Object** column is therefore renamed **Complement** and a **Complement Operator** column is inserted before it, indicating the relationship of the complement to the act or relation. The Complement Operator column mostly consists of prepositions, but can occasionally include other word types.

For instance, the "withdraw" act above can be extended as follows:

Subject	**Predicate**	**Complement operator**	**Complement**
Client (*human actor*)	Withdraws (*act*)		Cash (*informational object*)
		via	ATM (*physical object*)
		from	Account (*conceptual object*)
		at	ATM location (*place*)
		using	ATM Card (*informational object*)

The *client-account* relation above can be extended as follows:

Subject	**Predicate**	**Operator**	**Complement**
Client *(human actor)*	Has *(relation)*		Accounts *(conceptual object)*
		with	Bank *(institutional actor)*

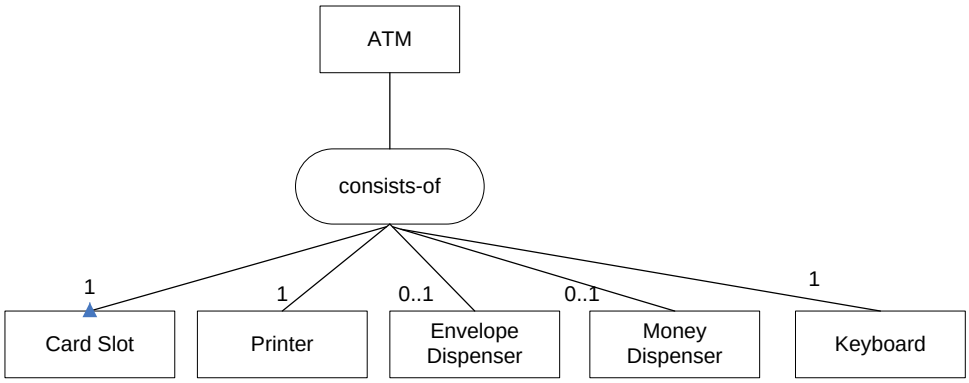
The model phrase can be further extended by adding a **complement qualification** column. This column qualifies the relation between the predicate and the complement. It can be as user-friendly (e.g. *only one*, *one-to-many*, *not more than 50*, *optional one*) or as precise (e.g. *1*, *1..n*, *<=50*, *0..1*) as needed for the analysis situation. For instance:

Subject	**Predicate**	**Operator**	**Qualification**	**Complement**
Client (*human actor*)	Has (*relation*)		0..Many	Accounts (*conceptual object*)
		with	1	Bank (*institutional actor*)
Branch (*institutional actor*)	Has (*relation*)		Only 1	Manager (*human actor*)
Branch (*institutional actor*)	Has (*relation*)		1..Many	Tellers (*human actor*)

Another example is where the relations between an ATM and its components can be described as follows:

Subject	**Predicate**	**Operator**	**Qualification**	**Complement**
ATM *(physical object)*	Consists-of *(relation)*		1	Card slot
			1	Printer
			0..1	Envelope dispenser
			0..1	Money dispenser
			1	Keyboard

This information can also be represented graphically as follows:



The relations between entities can be used to determine the granularity of the description used. For instance, in the beginning of analysis or on a higher level of specification, only the ATM can be addressed, e.g. “Place ATM card into ATM”. As more detail is uncovered and the components of an ATM become defined, that statement (although true) can be specified in more detail, e.g. “Place ATM card in ATM – card slot”.

Model phrases are joined together to form **model sentences**. The **Link** column is added and used to link model phrases together into sentences. For instance, the following is one model sentence consisting of four model phrases:

Link	Subject	Predicate	Operator	Qualification	Complement	Role type
When	Client	Inserts			ATM card	Event
			Into		ATM card reader	
If	PIN	Is-equal-to			PIN	Condition
			In		Client file	
Then		Display			ATM options menu	Action
			On		ATM screen	
Else		Display			Error message	Action

The structure entities making up a model phrase are as follows (see Figure 7-6):

- The **link** identifies the relationships between actions. The most common links between actions are *sequence* (actions following one another), *repetition* (actions forming a loop), *decision* (actions dependent on some condition) and *concurrency* (actions executed concurrently).
- The **subject** of the relationship. It refers to the base entity (*actor* or *object*) which is the main party in the model phrase.
- The **predicate** of the model phrase. It describes the action taking place or the type of relationship between the subject and the object. The predicate can play any of the following role types:
  - **Events** – Describe external triggers that cause agents to initiate actions or cause the state of things to change. For example, an important event type is the **timer event**, where either absolute time, like “on 2 August 2007”, or relative time, like “at month–end” and “at the end of the day”, cause actions to take place.
  - **Conditions** – Describe conditions, as part of a bigger model sentence, affecting later action’s steps, mostly action phrases. For example, “if the reorder level goes below 20”.
  - **Actions** – Describe how agents (human and non-human) perform work needed to reach the objectives of one or more individual, institutional or artificial agents. For example, “client places order”.
  - **Relationships** – Describe static relationships between subject and complement.

- The optional **complement operator** of the model phrase. It identifies the relationships of the various complements (direct object, indirect object, other complements) involved in the model phrase. Complement operators are mostly prepositions such as “for”, “in”, “from” and “to”.
- The optional **complement qualifier** of the model phrase. It identifies qualifications that must be placed on the complement. The most common object operators are multiplicity indications like “one or more” and “one only”.
- The **complement** of the action refers to the base entities (Actor or Object) that are affected by the predicate of the model phrase. It can also refer to the property of a base entity.

The ATM example can be defined formally by firstly defining every base entity and its relationships (see Table 7-2) and then defining all the model blocks and their corresponding events, actions and conditions (see Table 7-3 and Table 7-4).

Link	Subject	Predicate	Operator	Qualification	Complement
	ATM model	Is-a-type-of			<<Model>>
	ATM back-end model	Is-a-type-of			<<Model>>
	Deposit money	Is-a-type-of			<<Model Block>>
	Withdraw money	Is-a-type-of			<<Model Block>>
	Login client	Is-a-type-of			<<Model Block>>
	Client	Is-a-type-of			<<Human Actor>>
	Client	Is-a-type-of			<<Informational Object>>
	ATM	Is-a-type-of			<<Artificial Actor>>
	ATM card	Is-a-type-of			<<Informational Object>>
	Card slot	Is-a-type-of			<<Physical Object>>
	Printer	Is-a-type-of			<<Physical Object>>
	Envelope dispenser	Is-a-type-of			<<Physical Object>>
	Money dispenser	Is-a-type-of			<<Physical Object>>
	Envelope receiver	Is-a-type-of			<<Physical Object>>
	ATM back-end model	Has-member			Deposit money
	ATM back-end model	Has-member			Withdraw money
	ATM back-end model	Has-member			Login client
	Client (the actor)	Has		0..n	ATM card
	Client (the actor)	Has		0..n	Account
	ATM	Has		1	Card slot
	ATM	Has		0..1	Printer
	ATM	Has		0..1	Envelope dispenser
	ATM	Has		0..1	Money dispenser
	ATM	Has		1	Envelope receiver

Table 7-2: ATM example – relationships

These relationships can also be represented in graphical format (see Figure 7-7 for partial representations).

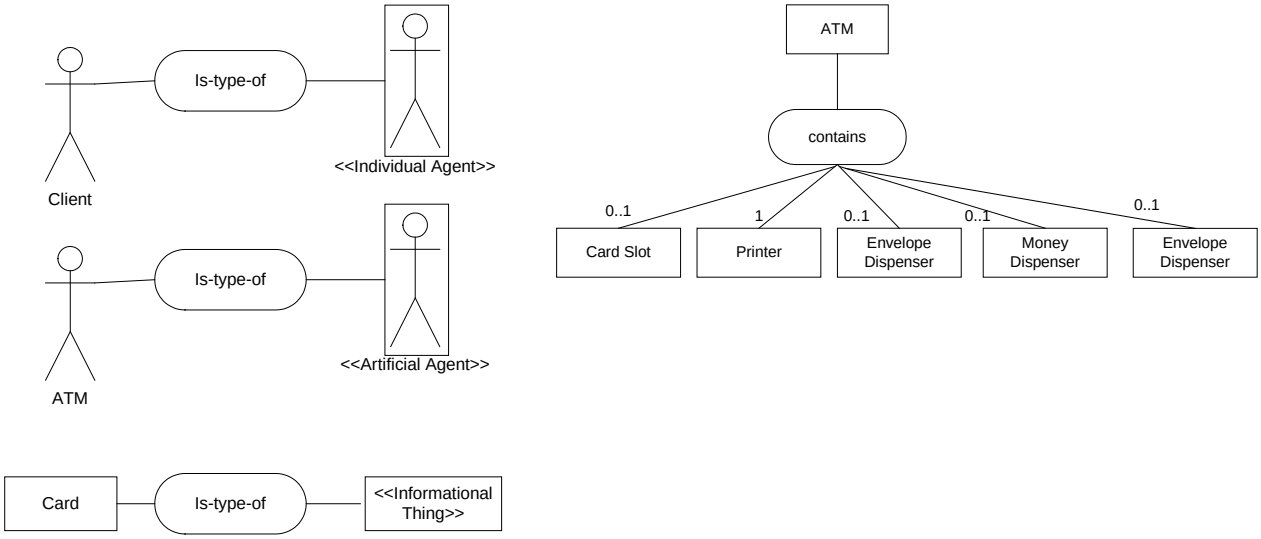


Figure 7-7: ATM example of relationships (graphical representation)

No	Link	Subject	Predicate	Operator	Qualification	Complement
1	When	Client	Inserts			ATM card
				Into		Card slot
2		System	Validates			ATM card
3	If	ATM card	Is			Invalid
4		System	Displays			Error message “Invalid card”
5	End if	System	Returns			ATM card
6		System	Displays			Login screen
7		Client	Enters			Pin
				On		Login screen
8	If	Pin	NOT (is-equal-to)			Pin
				On		Client
9		System	Displays			Error message
10	End if	System	Goes-to		6	Model block number

Table 7-3: ATM example of action – login to ATM

No	Link	Subject	Predicate	Operator	Qualification	Complement
1	When	Client	Selects			Withdraw money
				From		Select option screen
2		System	Requests			Account -- number
3		Client	Provides			Account -- number
4		System	Requests			Withdraw amount
5		Client	Provides			Withdraw amount
6		System	Updates			Account – available balance
7		System	Updates			ATM balance
8		System	Prints			Receipt
9		System	Dispenses			Money

Table 7-4: ATM example of action – withdraw money

These actions can also be represented in graphical format (see Figure 7-8 for partial representations).

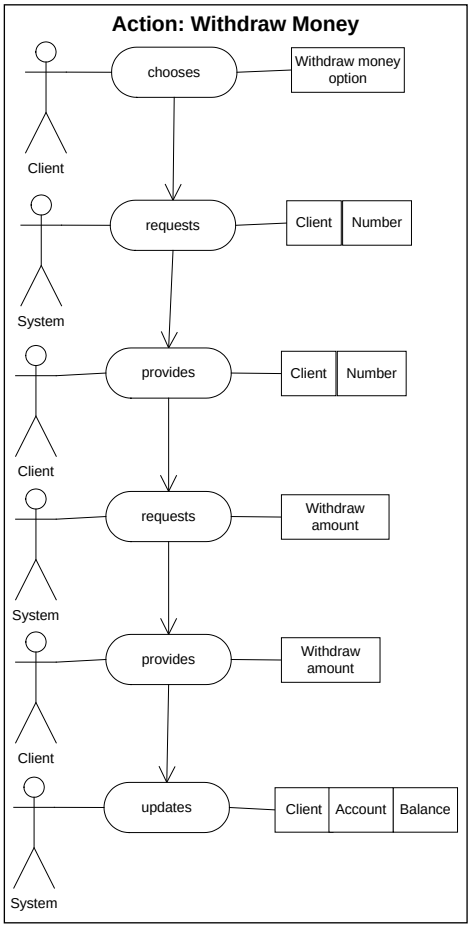


Figure 7-8: ATM example – withdraw money (graphical representation)

7.3.3 Role entities

7.3.3.1 Role entities overview

Every type of base and structure entity has a specific meaning depending on its use in a model sentence or phrase. These meanings are specified by the role entities. Role entities can be divided into four categories:

- **Subject role entities** identify the meanings that the subjects of acts or relations can have, namely **agent**, or **zero**, for instance:

Link	Subject	Subject Role	Predicate	Operator	Qualification	Complement
When	Client	Agent	Inserts			ATM card
				Into		ATM card reader
	Client	Zero	Is			Dormant

- If the subject assumes the **agent** role, it is actively involved in performing some action on an object, optionally involving other objects.
- If the subject assumes the **zero** role, it indicates a specific state for the subject.

- **Predicate role entities** identify the meanings that the predicate of acts or relations can have (note that the predicate on its own does not always fully define its role). In many cases, other parts of the modelling phrase, especially the *link*, are needed to define the predicate's role. In that sense the role of the predicate can be seen as the role of the whole modelling phrase). For instance:

Link	Subject	Predicate	Predicate Role	Operator	Qualification	Complement
When	Client	Inserts	Event			ATM card
				Into		ATM card reader
If	PIN	Is-equal-to	Condition			PIN
				In		Client file
Then		Display	Action			ATM options menu
				On		ATM screen
Else		Display	Action			Error message
	Client	Has	Relationship		1..n	Account

- **Actions** indicate dynamic relations (acts) between entities. Actions can be one of the following:
  - **Transformation** or processing of one entity into another, either by **assembly** or by pure transformation
  - **Transportation** or distribution of an entity from one place to another**,** including **arranging** objects in new patterns
  - **Storing** or housing an entity in a place
  - **Exchanging** or trading an entity for another entity of value
  - **Control** or regulation of an entity
- **Relationships** indicate static relations between entities. Relationships are mainly one of the following:
  - **Association,** where one entity is associated with another entity
  - **Property,** where one entity owns another entity
  - **Instance,** where one entity is an instance of another entity
  - **Recursion,** where an entity is associated with itself

  - **Aggregation** and **composition,** where an entity consists of other entities
  - **Inheritance,** where one entity is a type of another entity
  - **Location,** where one entity is located in a specific relationship to another location, e.g. “above”, “in”, “below”
- **Events** indicate acts that trigger other acts.
- **Conditions** indicate relations between entities that either allow or disallow actions from taking place.

- **Complement role entities** identify the meanings that the entities described in the complement of acts or relations can have. These complement roles can be grouped into the following categories (see Table 7-5 for examples):
  - **Basic**
    - **Patient** is an entity that is directly affected or effected (produced) by an action.
    - **Zero** is an entity involved in a relation.
  - **Object (or what)**
    - **Instrument** (or **tool**) is an entity used to affect an action.
  - **Location-related (or where)**
    - **Location** is a conceptual object of type of place where an action takes place or where an entity is located.
    - **Source** is an actor from which an entity originates.
    - **Direction** is an actor to which an entity moves.
    - **Path (or route)** is a conceptual object describing the way an entity moves or is oriented.
  - **Time-related (or when)**
    - **Speed** is a conceptual object describing how fast an entity moves.
    - **Frequency** is a conceptual object describing how often an action takes place in a certain period of time.
    - **Time point** is a conceptual object of type of time describing a specific point in time.
    - **Duration** is a conceptual object of type of time describing two points in time during which an action takes place.

- **Stakeholder (or who)**
  - **Beneficiary** is the actor who benefits or is interested in an action.
  - **Receiver** is the actor who receives something as a possession as a result of an action.
  - **Company** is the actor who takes part in an action with the agent.

Link	Subject	Predicate	Operator	Qualification	Complement	Complement Role
When	Client	Inserts			ATM card	Patient
			Into		ATM card reader	Instrument
If	PIN	Is-equal-to			PIN	Association
			In		Client file	Location
Then		Display			ATM options menu	Patient
			On		ATM screen	Location
Else		Display			Error message	Patient
	Client	Has		1..n	Account	Zero

Table 7-5: Examples of complement roles

The relationships between base, structure and role modelling entities can be described by the following overview:

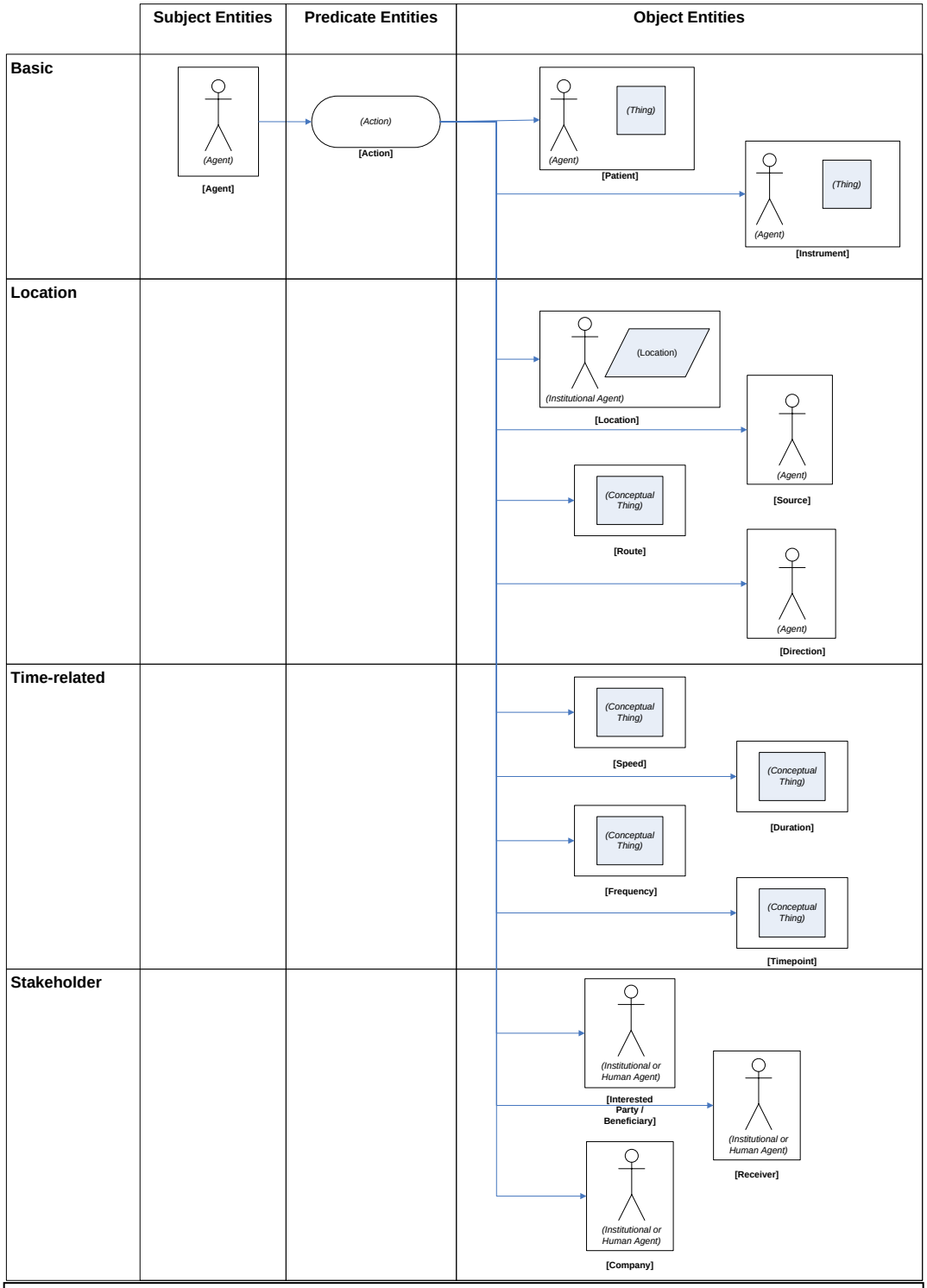
Both **institutional actors** and information systems (which are specific types of **artificial actor**) consist of the following elements and relationships:

- **Events** that trigger **actors** to initiate **actions** if certain **conditions** are met.
- **Actions** involve **things** and other **actors.**
- **Actions** and **things** that can cause **events**.

For instance, the ATM subsystem can be described in terms of the modelling entities as follows:

- The **event** *client needs cash* triggers **actor** *client* to initiate **model block** *withdraw money*.
- The *withdraw money* **model block** involves the following **objects**: *money, printing paper, ATM machine, ATM card, PIN number, client account*, and the following **actors**: *banking system, network system*.

- The *withdraw money* **model block** can cause **event** *ATM money depleted*, which in turn will trigger an **actor** *ATM administrator* to initiate **model block** *replenish ATM money*.
- Instead of waiting for the ATM's money to get depleted, the **event** *ATM balance is-below ATM money threshold* or the **event** *every third day* can trigger **actor** *ATM administrator* to initiate **model block** *replenish ATM money*.
- The change of state in **object** *ATM machine* can cause **event** *ATM broken* to occur.
- Every **object** *ATM machine* is allocated to **place** *ATM location*.



Note: Base entities are given in brackets and italics while role entities are given in square brackets and bold.

Figure 7-9: Role entities related to their corresponding base entities

In the next two sections, the main predicate role entities, action and relationship, are discussed in more detail.

7.3.3.2 Action

(a) Overview

The base entity **act** (e.g. *withdraw*) is the singular verb providing the predicate for the role entity **action** (e.g. *client withdraws money*) or **event** (e.g. *when client withdraws money*). Therefore a complete modelling phrase is needed to describe an action, and this is the subject of this section.

Verbs include any action that can be done by humans, such as walk, sort, write, withdraw money and apply for leave; any action that can be done by a computer system, like print statement and save client details; as well as any action that can be done by an organisation, like grant credit, create policy and appoint employees.

In business terms, verbs can be any of the functions of the enterprise, e.g. general management, purchasing, production, marketing, finances, human resources, information and public relations; the conversion processes of input, processing and output (e.g. taking raw material and creating a finished product); any services provided by an organisation; and the basic processes of extraction (agriculture and mining), manufacturing, transport, warehousing, services, provision (wholesale and retail).

In computer and programming terms, actions are related to the control structures defined by structured programming: sequence, selection (IF and CASE), repetitions (any loops) and modules (functions and subroutines). It is also related to concurrently running programs or threads.

Transactions are specific types of actions performed on information where **all** of the constituent actions must execute successfully as a single unit of work, or **none** must take place. If any constituent action fails, all actions already executed must be rolled back. Transactions must be atomic (seen as one unit), consistent (will not leave system in an illegal state), isolated (changes by other transactions must not influence this one) and durable (results of committed transactions must survive permanently).

An **action** can have three possible outcomes (following the BPMN technique (OMG, 2009:62–63)):

- **Successful completion**
- **Failed completion cancelled**: The activities inside the action will be subjected to the cancellation actions, which could include rolling back the process and compensation for specific activities.
- **Hazard**: This means that something went terribly wrong and that a normal success or cancel is not possible. The activity is interrupted with no rollback and without compensation.

Some activities produce complex effects or specific outputs. If the outcome is determined to be undesirable by some specified criteria (such as an order being cancelled), then it will be necessary to "undo" the activities. There are three ways in which this can be done:

- Restoring a copy of the initial values for data, thereby overwriting any changes.
- Doing nothing (if nothing has changed because the changes have been set aside until a confirmation).
- Invoking activities that undo the effects, also known as compensation. For example, an action that charges a buyer for some service and debits a credit card usually needs a separate activity to counter the effects of the initial activity.

(b) Components

At the highest level is a group of actions, combined in a model block, consisting of a set of **action**-related model phrases and sentences (i.e. the predicate role entities events, conditions and actions). For instance, all the processes in *run payroll process* form a model block, which can be divided into specific actions like *system calculates deductions*. See Figure 7-10.

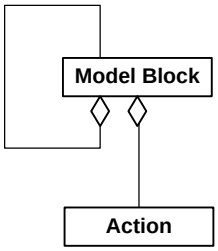


Figure 7-10: The composite relationships between actions

Executable model blocks can be initiated or triggered by some event. Then it corresponds closely to a use case in UML. Executable model blocks are normally named in one of two ways:

- With a verb followed by some object (noun) and other descriptive phrases (optional), for instance the following:

  - *Place order* (when the event is *client phones in*)
  - *Handle late returns* (when the event is *expected return date-time occurred without any return*)
  - *Conduct interview* (when event is *time for interview occurred*)

- With a noun representing the object, followed by a noun made from the corresponding verb, for instance the following:

  - *VAT calculation* (when the event is *stored module called by another action*).

In model blocks, actions follow other actions mostly sequentially, but also with decision, repetition and concurrency structures creating alternative paths. Actions are normally not named, but sometimes numbered if there is a need to refer to a specific action.

(c) Categories

Some of the categories into which actions can be placed are as follows:

- **Automated** actions are completely executed by some device or information system.
- **Semi-automated** actions are done by humans with assistance from some device or information system.
- **Manual** actions are executed entirely by humans.

The concept of action in this model includes all related action types like processes, functions, activities, procedures and use cases.

(d) Properties

Transformation actions can have two meanings. Firstly, it means transforming an entity from one state to another, e.g. washing a car and training people. Secondly, it means assembling something. Here a new entity "emerges" when a number of constituent entities are organised together, for instance, mixing the ingredients of a cake, setting up a project team and assembling a car from its parts.

Transformation actions have the following fundamental properties:

- The characteristic **input** identifies the fundamental entities that undergo the transformation. In assembling the input, entities are the "ingredients" of the output entity, and then a quantity and unit-of-measure must also be provided, e.g. 3 kg flour.
- Every input has an **input type** that identifies what sort of entity the input is. Input type can be any one of the modelling constructs identified (agent, thing, action, location, event).
- The characteristics **state before** and **state after** identify the state of the output entity before and after the action.
- The **output** indicates the resultant entity after the transformation. Note: the combination of the properties **action** and **output** typically provides the name of the action, e.g. bake cake, train people, wash car, produce report.
- The **output type** identifies what sort of entity the output is. Output type can be any one of the morphological constructs identified (agent, thing, action, location, event).

Examples of non-assembling transformation actions and their properties are as follows:

Example / Property	1	2	3
	Wash car	**Train people**	**Produce report**
Input	*Car, water, detergent*	*People*	*Transaction information*
Input type	*Physical thing*	*Human agent*	*Information*
State before	*Dirty*	*Untrained*	*Unsummarised*
State after	*Clean*	*Trained*	*Summarised*
Output	*Car (same thing)*	*People (same agent)*	*Monthly report*
Output type	*Physical thing*	*Human agent*	*Information*
Tools and material	*Cleaning material*	*Whiteboard, stationery*	*Calculator, computer*
Instructions/ rules	*Washing instructions*	*Training procedures*	*Monthly report procedure*
Participating agents	*Washer*	*Trainer*	*Manager*
Locations	*Wash area*	*Classroom*	*Manager's office*
Events	*Operation time, drying time*	*Course schedule, registration period*	*Month-end, report deadline*

The first example can be represented with the following model sentence:

Subject	Predicate	Operator	Qualification	Complement
Washer	Washes			Car
		Using		Cleaning material
		Following		Washing instructions
		In		Wash area

Examples of assembling transformation actions and their properties are as follows:

Example / Property	1	2	3
	Bake cake	**Set up project team**	**Assemble car**
Input	Ingredients • *Flour, 1 kg* • *Milk, 500 ml* • *Sugar, 50 mg*	Staff • *Project scope* • *Project manager, 1 ea* • *Business analysts, x ea* • *Systems analyst, y ea* • *Developers, z ea*	Parts • *Wheels, 4 ea* • *Engine, 1 ea* • *Body, 1 ea*
Input type	*Physical things*	• *Information* • *Human agents*	*Physical things*
State before	*Unbaked*	*Not set up*	*Unassembled*
State after	*Baked*	*Set up*	*Assembled*

Example / Property	1	2	3
	Bake cake	**Set up project team**	**Assemble car**
Output entity	*Cake*	*Project team*	*Car*
Output type	*Physical thing*	*Institutional agent*	*Physical thing*
Tools	*Oven, timer*	*Project management software*	*Physical tools, robots*
Instructions	*Recipe*	*Project management procedures*	*Assembly instructions*
Participating agents	*Baker, health inspector*	*Project manager, project steering committee, stakeholders*	*Welders, fitter and turners*
Locations	*Kitchen, pantry*	*Office*	*Assembly line*
Events	*Bake time, cooling time, mix setting*	*Project initiation*	*Assembly times, shift time*

The first example can be represented with the following model sentence:

Subject	Predicate	Operator	Qualification	Complement
Baker	Bakes			Cake
		From	1 kg	Flour
			500 ml	Milk
			50 mg	Sugar
		Following		Recipe
		In		Kitchen
		Using		Oven
				Timer

Transport actions involve taking entities from one location and moving them to another location. For instance, it involves moving fruit during the packaging process, transporting passengers by bus and sending mail by post. It involves either one entity or a number of entities that can be seen as one, like a ship container.

Examples of transport actions for physical things, informational things (which are physical and not electronic) and human agents (who are also physical in this sense) and their properties are as follows:

Example / Property	1	2	3
	Distribute product	**Transport passengers**	**Send mail**
Transported entity (input and output)	*Product*	*Passenger*	*Mail (physical)*
Entity type	*Physical thing*	*Human agent*	*Informational thing*
Location from	*Packaging area*	*Suburb X*	*Agent A's post office*

Example / Property	1	2	3
	Distribute product	**Transport passengers**	**Send mail**
Location to	*Customer*	*Suburb Y*	*Agent B's postal address*
Transport medium	*Conveyor belt, delivery van, trains*	*Bus*	*Postal vans*
Instructions/rules	*Distribution procedures*	*Municipal transport by-laws*	*Postal delivery procedures*
Participating agents	*Truck drivers, packers*	*Bus driver*	*Post sorters, truck drivers, postmen*
Events	*Order arrival*	*Bus schedule*	*Daily delivery time*

The second example can be represented with the following model sentence:

Subject	**Predicate**	**Operator**	**Qualification**	**Complement**
Bus driver	Transports		Many	Passenger
		From	X	Suburb
		To	Y	Suburb
		Via		Bus
		Following		Municipal transport by-laws

Examples of transport actions for conceptual things (mostly electronic) and their properties are as follows:

Example / Property	1	2
	Broadcast TV show	**Send email**
Transported entity (input and output)	*TV show*	*E-mail*
Entity type	*Information*	*Information*
Location from	*Broadcasting towers*	*Originating email address*
Location to	*Receiving TV*	*Destination email address*
Transport medium	*Television signal*	*Email network*
Instructions/rules	*TV broadcasting regulations*	*Postal delivery procedures*
Participating agents	*Satellite service provider*	*Sender, service providers*
Events	*TV show published broadcast times*	*Email failure (causing resend)*

The second example can be represented with the following model sentence:

Subject	Predicate	Operator	Qualification	Complement
Sender	Sends			E-mail
		From		Originating email address
		To		Destination email address
		Via		Email network
		Involving		Service providers

Transport or **distribute** can also be interpreted as arranging entities. For instance, arrange products on the shelves, sort fish and pack new library books. Fundamentally, it means moving many entities from one place to another using some sort of arrangement category, e.g. by size, by price or by type.

Examples of arranging actions (physical things) and their properties are as follows:

Example / Property	1	2	3
	Arrange products	**Sort fish**	**Pack library books**
Arranged entity (input and output)	*Products*	*Fish*	*Library books*
Arranging category	*By product categories*	*By type and size*	*By Dewey category and author name*
Entity type	*Physical thing*	*Physical thing*	*Information*
Location	*Display shelf position*	*Market, sorting tables and baskets*	*Library shelf position*
Instructions	*Packing instructions*	*Sorting procedure*	*Dewey system manual*
Participating agents	*Packer*	*Sorter*	*Librarian*
Events	*Display shelves empty, product delivery, shelves become unorganised*	*Daily market times*	*New book arrival*

The third example can be represented with the following model sentence:

Subject	Predicate	Operator	Qualification	Complement
Librarian	Packs			Library book
		At		Library shelf position
		Using		Dewey system manual
		By		Dewey category
				Author name
		On		New book arrival

Examples of arranging actions on conceptual things and their properties are as follows:

Example / Property	1	2
	Arrange fields on display screen	**Sort names**
Arranged entity (input and output)	*Textboxes, labels, buttons and other screen control*	*Names*
Entity type	*Information*	*Information*
Arranging category	*Typical HCI standards, left to right, top to bottom*	*By surname, then name both ascending*
Location	*On display screen (relative position)*	*In spreadsheet cells A1 – B354 (relative position in list)*
Instructions	*Arranging instructions*	*Sorting instructions*
Participating agents	*Screen designer*	*Spreadsheet user, spreadsheet software*
Events	*Screen design*	*Name received*

The second example can be represented with the following model sentence:

Subject	Predicate	Operator	Qualification	Complement
Spreadsheet user	Sorts		Many	Name
		By	Ascending	Surname
		Then by	Ascending	Name
		From	A1	Cell
		To	B354	Cell
		Using		Spreadsheet software
		Following		Sorting instructions

Storing or housing actions means taking entities and keeping and maintaining them in a specific location.

Examples of storing actions (for physical entities) and their properties are as follows:

Example / Property	1	2	3
	House prisoners	**Store medicine**	**Store books**
Stored entity	*Prisoners*	*Medicine*	*Books*
Entity type	*Human agent*	*Physical thing*	*Informational thing*
Storage location	*Prison, cells*	*Pharmacy's storeroom, shelf position*	*Library shelves*
Entry event	*When prisoner arrives with valid documentation*	*When valid delivery is received from supplier*	*When ordered books arrive*
Exit event	*When valid release authorisation is received*	*When valid dispensing instructions are received, when medicines have expired*	*When valid book lending takes place, when book is sent to another library or when book is written off*
Maintenance rules	*Prisoners have varying rights, e.g. visitation*	*Certain medicines must be kept refrigerated, or locked*	*Book pages and covers are restored when damaged*
Instructions/rules	*Prison rules and regulations*	*Medical rules and regulations concerning medicine*	*Library rules and regulations*
Participating agents	*Wardens, parole officers*	*Pharmacist*	*Librarian*
Other events	*Sentence length, prisoner becomes sick or dies*	*Expiry date, medicine damaged*	*Book damaged*

The second example can be represented with the following model sentence:

Subject	Predicate	Operator	Qualification	Complement
Pharmacist	Stores			Medicine
		In		Store room
		On		Shelf
		Until		Expiry date
		Following		Medicine storage regulations

Examples of storing actions for conceptual entities and their properties are as follows:

Example / Property	1	2
	Register client	**Archive reports**
Stored entity	*Client information*	*Report*
Entity type	*Information*	*Information*
Storage location	*Sales database, client table*	*Archive database*
Create rules	*If user has necessary create authorisation and client information is valid, then create record*	*When report older than 1 year, archive*
Read rules	*If user has necessary read authorisation, then read client record*	*If user has necessary read authorisation, then read archived report*
Update rules	*If user has necessary update authorisation and client information is valid, then update record*	*N/a*
Delete rules	*If user has necessary delete authorisation and client balance = 0, then delete or archive record (note: archive = logical delete)*	*If user has necessary delete authorisation or archived report older than 10 years, then delete archived report*
Instructions	*CRM rules*	*Archiving rules*
Participating agents	*Client, user, CRM system*	*User, archiving system*
Events	*Client registers, client changes any info*	*Archive expiry times*

The first example can be represented with the following model sentence:

Link	Subject	Predicate	Operator	Qualification	Complement
If	User	has			Create authorisation
Then	User	Register			Client
			In		Client database
			To		Suburb
			Using		CRM system
			Following		CRM rules

Exchange actions means swapping one entity (agent, thing, location) for something of value (thing, specifically money – an informational thing).

Examples of exchange actions and their properties are as follows:

Example / Property	1	2	3
	Buy plot	**Rent video**	**Exchange car for caravan**
Entity	*Plot*	*Video*	*Car*
Entity type	*Location*	*Informational thing*	*Physical thing*
Value entity	*Money*	*Money or contract points*	*Caravan*
Value entity type	*Informational thing*	*Informational thing*	*Physical thing*
Participating agents	*Buyer, seller, bank, lawyers, agents, deeds office*	*Video shop, client, shop assistant*	*Car owner, caravan owner*
Location	*Estate agent*	*Video shop*	*Garage*
Instructions	*Property laws and home loan policies*	*Video shop regulations*	*Common law*
Events	*Cooling off period, offer deadline*	*Contract expiry*	*Swap transaction*

Control actions mean ensuring that an entity is in a certain state.

Examples of control actions and their properties are as follows:

Example / Property	1	2	3
	Authorise employee salary increase	**Credit-check client**	**Test application program**
Entity	*Employee salary increase*	*Client*	*Application program*
Entity type	*Information*	*Human agent*	*Artificial agent*
State before	*Not authorised*	*Unchecked*	*Untested*
State after	*Authorised*	*Checked*	*Tested*
Success criteria	*If performance appraisal mark > 4*	*If client not black-listed at credit bureau*	*Compliance with test script criteria*
Success action steps	*Increase salary, send letter to employee*	*Change client's credit limit, send letter to client*	*Accept program*
Failure action steps	*Send letter to employee*	*Refer client to management*	*Return program to developer to fix problems*
Participating agents	*Employee, manager, HR department*	*Client, credit controller, credit bureau*	*Developer, tester*
Events	*Yearly salary increase period*	*Credit application by client*	*Testing dates in project plan*

The first example can be represented with the following model sentence:

Link	Subject	Predicate	Operator	Qualification	Complement
When		Is		Yearly	Salary increase period
If	Employee performance appraisal mark	Is-greater-than			4
Then	Manager	Authorise			Salary increase
			For		Employee
And then	Manager	Send			Letter
			To		Employee

7.3.3.3 Relationships

(a) Overview

Relationships show how different base entities are related to each other. Similarly to actions, it takes a complete model phrase to describe a relationship. Relationships can either be static or dynamic.

Static relationships are those that describe the state of a system before or after an action or event has taken place. It can involve any base entity except **act** (an act always involves an active verb and implied action and can therefore not be a static relationship). It is important to realise that a static view can change after every action/event in the system. This implies that when a static view is modelled, it will illustrate some typical or desired state. For instance, before a factory assembles a product, there are no relationships between any of the constituent parts. After the assembly, the typical "bill of material" relationship (or aggregation relationship) exists between the various parts.

Some of the types of static relationships between base entities that can be identified are as follows:

- **Association** – where one entity is associated with another. For example, a client (human actor) can have contracts (informational objects). This is mostly indicated by the auxiliary verb *has*.

- **Property** – where one entity is the property of another entity, for example *name* is a property of *client*.
- **Instance** – the relationship between a type and the instances of that type. For example, student *John* with student number *234518* is an instance of type *student* (human agent). Mostly indicated by the verb phrase *is-a*.
- **Recursion** – where an entity is related to itself. For example, a course (informational object) is related to other courses as prerequisite courses. Mostly indicated by the auxiliary verb *has*.
- **Aggregation and composition** – where one entity consists of other sub-entities. For instance, a car (physical object) consists of an engine (physical object), a chassis (physical object), a body (physical object) and 4 wheels (physical objects). It has to do with breaking things down into their constituent parts. This breaking down carries on until one has primitive entities – entities that cannot be broken down any further. Note that this can involve entities of the same or different types. For instance, while the car (physical object) above consists of other physical objects, a company (institutional actor) can consist of departments (institutional actors) but also of staff (human actors), buildings (places) and equipment (physical objects). Composition is a "stronger" relationship than aggregation and must also comply with the following criteria: only one entity in the relationship represents the whole; the parts in the relationship exist only as long as the whole exists, and the whole is responsible for creating and destroying its parts; and a part may only belong to one whole at a time, but it can be attached to another whole (following UML). Aggregation is mostly indicated by the verb phrase *consists-of*.
- **Generalisation-specialisation/inheritance** – where a number of entities have commonalities in another entity. For example, student (human actor) and staff (human actors) can both be seen as persons (human actors). Mostly indicated by the auxiliary verb phrase *is-a-type-of*.

Action relationships define how actions and subactions are related to each other. Any action group (such as a process, procedure or program) can be divided into subactions (such as subprocesses, subprocedures and subprogram). These subactions, in turn, can be divided into subactions until they cannot be divided any more.

Action groups are mostly given specific unique names, so that they can be accessed and used as a single unit. For instance, the *Credit_Check* process, the *TaxCalculation* function, the *Promote Employee* procedure and the *Traditional Bread* recipe.

When actions are executed, their subactions and steps follow each other in a specific order to achieve the desired goal of the action. This implies that there is a specific relationship between the subactions of an action. Changing the relationships between the subactions (without changing the subactions themselves) will change the overall working of the action. This is called the control flow of the action. This control flow in an action can occur in a number of ways:

- **Non-concurrent execution:** The execution starts at the beginning step of an action and carries on until it reaches any of a number of possible end steps. Only one step at a time can be executed. For example, eating a meal.
- **Concurrent execution:** The execution starts at the beginning step of an action and carries on until it reaches any of a number of possible end steps. More than one step can be executed at the same time. The action is finished when the last of the concurrent steps are finished. For example, baking a cake (while base is baking, the chef makes the icing), running a project (many actions can be performed by different members at the same time).
- **Event-driven execution:** Any of the above two types of executions can take place. While it is executing normally, an external event can stop the normal flow and jump the control to specific actions based on the type of event. For example, a long printing job can be cancelled (the cancel instruction is an external event to the printing, stopping the normal printing steps and executing the stop printing steps).

The control flow between action steps can be classified as follows:

- **Sequence**: This is where the actions follow one another in sequence.
- **Jump:** This is where the next action is at a totally different point in the action flow. This point can go back to actions already executed or go to actions that have not been executed. The action that is jumped to must be identified in some way.

A jump is typically called a *go to* in programming, but *exit* structures, which jump out of a loop or module before it reaches the end, are also jumps. For example, “go to step 245”, “jump to the next paragraph” and “go and read the last page of the chapter”.

- **Decision:** This is where a condition can cause the flow to go onto two or more paths, depending on the value of the condition. In programming this is typically called *if… then… else* or *case*. For example, “if it is raining, use a wet barometer, otherwise use a dry barometer”; and “if your marks are above 75, your code is ‘A’, if between 50 and 75 your code is ‘B’, if below 50 your code is ‘C’”.
- **Repetition:** This is where a number of actions are executed repeatedly for a certain number of repetitions, while a certain condition is true or until a certain condition becomes true. In programming, this is typically called *for*, *do while* or *loop until*. For example, “give the chair four coats of paint”; “while you still feel strong, run” (otherwise walk); and “bake the bread until the crust is light brown”.
- **Action calling:** An action is a group of action steps and can be named and executed as if they are one action step. In programming, functions, procedures, subroutines and methods are actions.
- **Recursion:** This is action calling but where an action calls itself during its execution.

(b) Properties

Most relationships are binary (but can be extended to n-ary).

- **Entity 1 … n:** The morphological entities taking part in the relationship
- **Relationship main type:** Static or dynamic relationship
- **Cardinality:** The number of entities taking part in the relationship. Each relationship has cardinality. Typical cardinalities are one-to-one, one-to-many and many-to-many
- **Mandatory/optional:** Indicates if the entities involved in a relationship are mandatory or optional for the relationship. For instance, a client does not have to have contracts (i.e. it is optional), but a contract must have a client.

(c) Links to other base entities

Typical actor relationships are as follows:

Relationship	**Typical pattern**	**Relationship type**
Contract	Actor 1 has contract with Actor 2	Association
Agency/ membership	Institutional actor/artificial actor consists-of actors	Composition
Responsibility	Actor 1 has responsibility/obligation towards Actor 2 to-do action	Association
Commitment	Actor 1 has-commitment-with Actor 2 to-do action	Association
Obligation/duty	Actor 1 has-obligation-to Actor 2 to-do action	Association
Delegation	Actor 1 has-authority-from Actor 2 to-do action	Association
Beneficiary/ customer	Actor is-beneficiary-of action	Association
Actor/executor	Actor is-actor-of action	Association
Initiator	Actor is-initiator-of action	Association
Authorisation	Actor is-authorised-to-do action	Association
Prohibition	Actor is-prohibited-from-doing action	Association
Source	Actor is-input-provider-to action	Association
Destination	Actor is-output-receiver-from action	Association
User	Actor is-user-of thing	Association
Creator	Actor is-creator-of thing	Association
Controller	Actor is-controller-of thing	Association
Perceiver	Actor is-perceiver-of event	Association
Claim	Actor has-claim-on event	Association
Located	Actor is-located-at location	Association
Owner	Actor is-owner-of location/actor/thing/ action	Association

Typical place relationships are as follows:

Relationship	Typical pattern	Relationship type
Spatial relationships	Place spatial-relationship Place 1 For example: Place 1 is-left/right-of Place 2 Place 1 is-above/below Place 2 Place 1 is-behind/in-front-of Place 2 Place 1 is-inside/outside Place 2 Place 1 is-far/near Place 2 Place 1 is-touching Place 2	Association
	Agent/action/thing/place is-located-at place	Association
Subplace	Place 1 is-divided-into/consists-of Place 2, … , Place x	Composition
Containment	Place 1 contains agent/action/thing	Association

Typical object relationships are as follows:

Relationship	Typical pattern	Relationship type
Relation	Object is-related-to object/action/agent	Association
Classification	Object is-a-type-of/is-a object	Inheritance
Instance	Object is-an-instance-of object	Instance
Composition	Object consists-of object Object is-part/component-of object	Composition
Stuff	Object is-stuff-of object	Composition
Portion	Object is-portion-of object	Composition
Membership	Object is-member-of object	Composition

7.3.3.4 Conditions

Conditions are used in many of the action control structures discussed above. They will always either be true or false. Conditions have the following elements:

- **Operands:** The things compared
- **Relational operators:** These operators are used to compare the operands. The operators are typically *is equal to, =, is not equal to, <>, is greater than, >, is less than, <, is greater or equal to, >=, is less than or equal to, <=*.
- **Logical operators:** These operators combine operands together. The operators typically are: *not*, *and*, *or*, *xor*, *andalso*, and *orelse*.

- **Arithmetic operators:** If operands are numeric, they can be combined with other numeric operands. The operators typically are *plus, +, minus, -, multiply, x, divide, /, integer divide, \, modulus, to the power of*.
- **Brackets:** Brackets are used to enforce certain precedence. If brackets are not used, a default precedence will be used.
- **Functions:** Functions take input parameters and produce a specific answer. In calculating the truth value of the condition, the answer returned by the function is used. The available functions depend on the specific situation in which the condition is used, like the programming language, spreadsheet type and database type. Examples of functions are *Sin(x), Cos(x), Max(x_1 .. x_n), Log(y), DateDiff(Date1, Date2), NPV(rate, value1, value2), Today(), Average(x_1 .. x_n), IsNumeric(x)*.

Examples of conditions are the following:

- (Gross salary - tax) x rebate > 100
- NOT IsNumeric(x)
- Sin(x) + Cos(x) <= 0
- EOF = True (end of file is reached)

7.3.3.5 Role entity analysis

The role value of any entity is given by a combination of factors. The role value of an entity can be given by the meaning and lexical type of the word that describes the entity. For instance, the verb “sing” shows dynamic action, while the verb “sit” shows passive keeping of a position. The use of auxiliary verbs indicates in many cases a relationship or state and not an action, for instance, “the store is insured” and “the store has five rooms”.

The role value of an entity can also be given by categories of words. It implies that when an analyst encounters a word or combination of words implying a concept like transport, certain questions can be asked as a rule, such as what is transported, from

whom (the source), to whom (the destination), to where (location), via which route (path), using which transport medium (instrument)?

To illustrate how this type of analysis can be done, a number of examples will be given and analysed using the base entities as well as the role entities, and then these will be discussed. The base entity analysis is shown in brackets and italics, e.g. *(Physical thing)*, while the role entity analysis is shown in square brackets and bold, e.g. **[Instrument].**

Example 1 (see Table 7-6) below is a transformation action (a transformation or process action transforms one entity into another one). In this example, car parts are transformed into cars.

Subject	**Predicate**	**Operator**	**Qualification**	**Complement**
Car manufacturer (*Institutional actor*) **[Agent]**	Builds (*Act*) [**Action**]			Car (*Physical object*) [**Patient**]
		From		Car parts (*Physical object*) [**Patient**]
		In	Many	Factory (*Institutional actor*) [**Company**]
		In	Many	City (*Conceptual object – place*) [**Location**]
		Using	Many	Robot (*Artificial actor*) [**Instrument**]

Table 7-6: Base and role entity analysis, Example 1

Example 2 (see Table 7-7) below is a move action (a move or distribute action moves an entity from one location to another). In this example, parcels are transported from the USA to South Africa. Words indicating transport, like “send”, “move” and “drive” in a sentence are many times followed by the prepositions “to” and “from” (and sometimes “via”) or equivalents. The meaning is transporting something from one place to another place on some sort of route.

Subject	Predicate	Operator	Qualification	Complement
DHL USA (*Institutional actor*) **[Agent]**	Sends (*Act*) [**Action**]			Parcel (*Physical object*) [**Patient**]
		Involving		SA Parcels Company (*Institutional actor*) [**Company**]
		From		DHL USA client (*Institutional/human actor*) [**Source**]
		To		DHL SA client (*Institutional/human actor*) [**Destination**]
		In		South Africa (*Place*) [**Location**]
		Via		Dubai route (*Conceptual object*) [**Path**]
		Every		Day (*Conceptual object – time*) [**Time duration**]
		During	Working	Hours (*Conceptual object – time*) [**Time duration**]

Table 7-7: Base and role entity analysis, Example 2

Example 3 (see Table 7-8) below is a store or house action. In this example, new library books are stored.

Subject	Predicate	Operator	Qualification	Complement
Librarian (*Human actor*) **[Agent]**	Stores (*Act*) [**Action**]		New	Book (*Informational object*) [**Patient**]
		On		Shelf (*Physical object*) [**Location**]
		According to		Dewey classification system (*Conceptual object*) [**Manner**]
		At	50	Books (*Institutional/human actor*) [**Destination**]
		Per		Hour (*Conceptual object – time*) [**Duration**]

Table 7-8: Base and role entity analysis, Example 3

Example 4 (see Table 7-9) below is an exchange action. In this example, second-hand cars are sold.

Subject	**Predicate**	**Operator**	**Qualification**	**Complement**
Client (*Institutional actor*) **[Agent]**	Buys (*Act*) [**Action**]			Second-hand car (*Physical object*) [**Patient**]
		For		Selling price (*Conceptual object*) [**Patient**]
		At		Dealership (*Institutional actor*) [**Company**]

Table 7-9: Base and role entity analysis, Example 4

Example 5 (see Table 7-10) below is an example of a business rule. It indicates the operational hours of a bank.

Subject	**Predicate**	**Operator**	**Qualification**	**Complement**
Bank branch (*Institutional actor*) **[Zero]**	Is open (*relation*) [**Relationship**]	On	Working	Days (*Conceptual object – time*) [**Time point**]
		From	Opening	Hour (*Conceptual object – time*) [**Time point**]
		To	Closing	Hour (*Conceptual object – time*) [**Time point**]
Opening -- Hour (*Conceptual object – time*) [**Zero**]	Has-value			09:00
Closing -- Hour (*Conceptual object – time*) [**Zero**]	Has-value			16:00
Week day (*Conceptual object – time*) [**Zero**]	Has-value-domain	From		Monday
		To		Saturday

Table 7-10: Base and role entity analysis, Example 5

7.4 Conclusion

In this section, the integrative modelling technique developed as part of this research was explained in more detail. In summary, it consists of the following three levels of modelling entities:

- **Base entities** (the words of the modelling language), representing the real-world objects in the organisation and IS:

  - Human, institutional and artificial **actors** (intelligent things) that can act and make decisions.
  - Physical, conceptual (information, place and time) and informational **objects** (non-intelligent things).
  - Dynamic **acts** that actors can perform on objects and other actors.
  - Static **relations** between objects, between actors and between objects and actors.

- **Structure entities,** combining base entities and specific language constructions to form the phrases, sentences, models and views of the modelling language:

  - **Model phrases** are formed from the basic structure entities: **link, subject**, **predicate**, **complement operator**, **complement qualifier** and **complement.**
  - **Model sentences** are one or more model phrases that are linked together.
  - Named **model blocks** are groups of model phrases and sentences.
  - **Models** represent all available information on the real-world situation.
  - **Model views** represent one or more specific aspects of a model.

- **Role entities:** The roles and meanings all the other entities play in the modelling situation. The entities can be grouped as follows:

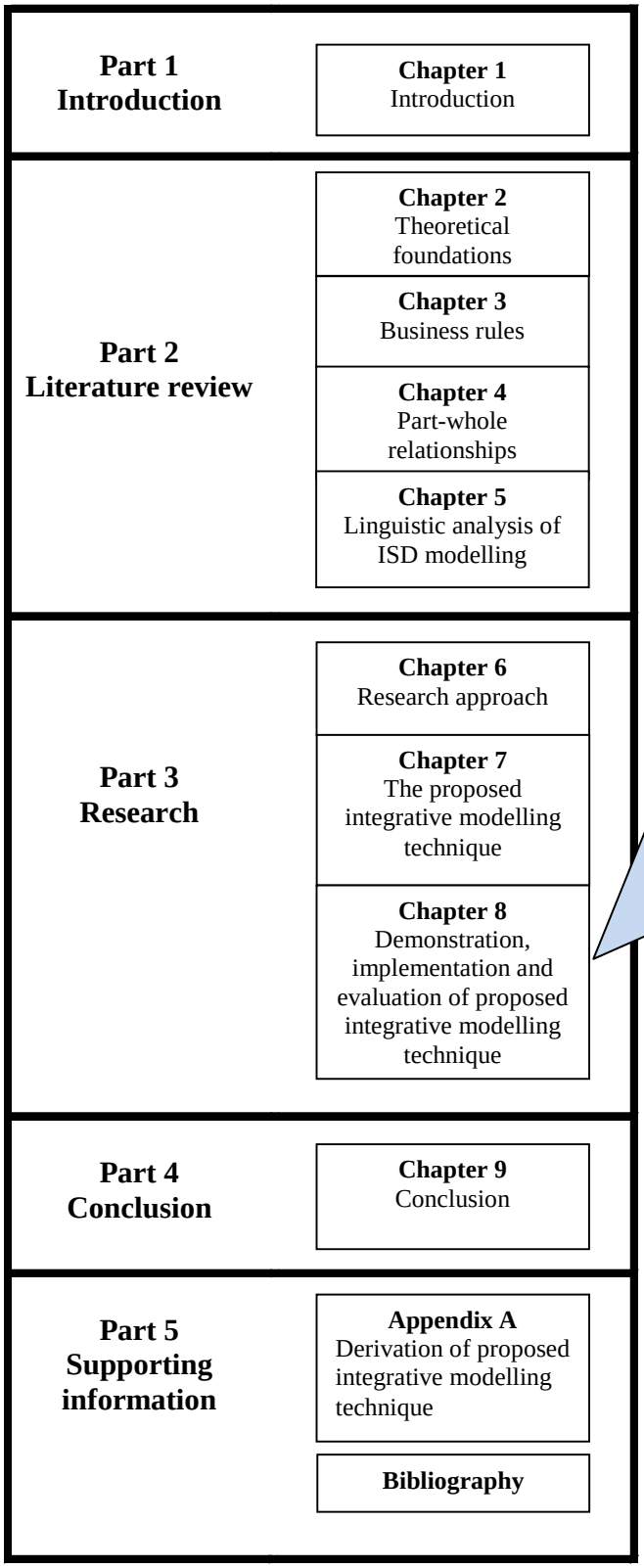
  - **Subject roles** of agents and zero
  - **Predicate roles** of events, conditions, actions and relationships
  - Basic, object-related, location-related, time-related and stakeholder-related **complement roles**

It should be clear that only three of the linguistic levels have been used. Although the pragmatic level of linguistics can add a lot to the usability of this modelling technique, it will only be considered in further research. In short, the structure of typical human-computer discourses can be developed as patterns of specific model phrases and

sentences. For instance, entering data will always involve a set of model phrases doing data validation first.

This modelling language aims to be **easy to use** by non-technical business users (being as close as possible to natural language), but at the same time to be **expressive enough** (by formalising the language with "just enough" structure) so that it can be translated to ISD modelling mostly procedurally (i.e. following a set of instructions without any need to do first-level analysis again). These requirements are evaluated in the next chapter.

8. Demonstration, implementation and evaluation of proposed integrative modelling language



8.1 Introduction
8.2 Case study
8.3 Evaluation per perspective
- **8.3.1 Perspective 1 – Business rules**
- **8.3.2 Perspective 2 – ISD modelling**
- **8.3.3 Perspective 3 – Requirements modelling using use cases**

8.4 Implementation of the technique as software
8.5 Linking the integrative technique back to existing ISD techniques
8.6 Conclusion

8.1 Introduction

The proposed integrative modelling technique is demonstrated and evaluated from a number of different perspectives to ensure that it actually can achieve its purpose of filling the gap between business and ISD modelling.

In section 6.4.2, the design objectives of this study were clearly stated. Therefore, for each of the following perspectives, the criterion for success must be that the technique is **easy to use** for a non-technical domain expert, but have **enough expressive power** so that it can be used to derive detailed ISD models.

- The **business rules perspective** ensures that the modelling technique can model any business rule (or any other rule for that matter). Business rules have been discussed in Chapter 3 in detail. In summary, business rules are either terms, facts, constraints (action assertions) or derivations, and the integrative modelling technique must be able to model all of these types of business rules.
- The **ISD perspective** ensures that the integrative modelling technique can be used to model any business aspect of ISD through all the phases of a typical SDLC. For the sake of this study, the following phases are assumed for a typical SDLC: requirement analysis, system design, application and program design, testing and implementation. Business modelling is applicable specifically to the phases of requirement analysis and user acceptance testing.
- The **requirements perspective** ensures that the most commonly used requirement modelling tool, use cases, can be modelled using the integrative technique.

8.2 Case study

The following case study is an extract of a bigger case study found in the Business Rules Group's document on business rules (Hay and Healy, 2000:D1–D8). They developed the case study together with a number of companies, specifically to cover all possible types of business rules. The case study is of such detailed information that it can also be used to illustrate how to analyse it with the purpose of developing an IS from it. The case study involves a car rental company called EU-Rent. For the purpose of referencing source statements in the case study, every sentence/paragraph

in the case study has been numbered in the version below, unlike the original case study.

Case study: EU-Rent car rentals

1. EU-RENT car rentals

1.1.1 EU-Rent is a car rental company owned by EU-Corporation. It is one of three businesses – the other two being hotels and an airline – that each has its own business and IT systems, but with a shared customer base.

1.1.2 Many of the car rental customers also fly with EU-Fly and stay at EU-Stay hotels.

2. EU-RENT business

2.1.1 EU-Rent has 1 000 branches in towns in several countries.At each branch, cars, classified by car group, are available for rental. Each branch has a manager and booking clerks who handle rentals.

2.2 Rentals

2.2.1 Most rentals are by advance reservation; the rental period and the car group are specified at the time of reservation. EU-Rent will also accept immediate ("walk-in") rentals if cars are available.

2.2.2 At the end of each day, cars are assigned to reservations for the following day. If more cars have been requested than are available in a group at a branch, the branch manager may ask other branches if they have cars they can transfer to him/her.

2.3 Returns

2.3.1 Cars rented from one branch of EU-Rent may be returned to a different branch. The renting branch must ensure that the car has been returned to some branch at the end of the rental period. If a car is returned to a branch other than the one that rented it, ownership of the car is assigned to the new branch.

2.4 Customers

2.4.1 A customer can have several reservations, but only one car rented at a time. EU-Rent keeps records of customers, their rentals and bad experiences, such as late returns, problems with payment and damage to cars. This information is used to decide whether to approve a rental.

3. EU-RENT BUSINESS RULES

3.1 External constraints

3.1.1 Each driver authorised to drive the car during a rental must have a valid driver's licence.

3.1.2 Each driver authorised to drive the car during a rental must be insured to the level required by the law of each country that may be visited during the period of rental.

3.1.3 Rented cars must meet local legal requirements for mechanical conditions and emissions for each country that may be visited during the period of rental.

3.1.4 Local tax must be collected (at the drop-off location) on the rental charge.

3.2 Rental reservation acceptance

3.2.1 If a rental request does not specify a particular car group or model, the default is group A (the lowest-cost group).

3.2.2 Reservations may be accepted only up to the capacity of the pickup branch on the pickup day.

3.3.3 If the customer requesting the rental has been blacklisted, the rental must be refused.

3.3.4 A customer may have multiple future reservations, but may have only one car at any time.

3.3 Car allocation for advance reservations

3.3.1 At the end of each working day, cars are allocated to rental requests due for pickup the following working day. The basic rules are applied by a branch:

3.3.1.1 Only cars that are physically present in EU-Rent branches may be assigned.

3.3.1.2 If a specific model has been requested, a car of that model should be assigned if one is available. Otherwise, a car in the same group as the requested model should be assigned.

3.3.1.3 If no specific model has been requested, any car in the requested group may be assigned.

3.3.1.4 The end date of the rental must be before any scheduled booking of the assigned car for maintenance or transfer.

3.3.1.5 After all assignments in a group have been made, 10% of the group quota for the branch (or all the remaining cars in the group, whichever number is lower) must be reserved for the next day's walk-in rentals. Surplus capacity may be used for upgrades.

3.3.1.6 If there are not sufficient cars in a group to meet demand, a free one-group upgrade may be given (i.e., a car of the next higher group may be assigned at the same rental rate) if there is capacity.

3.3.1.7 Customers in the loyalty incentive scheme have priority for free upgrades.

3.4 Walk-in rentals

3.4.1 The end date of the rental must be before any scheduled booking of the assigned car for maintenance or transfer.

3.4.2 If there are several available cars of the model or group requested, the one with the lowest mileage should be allocated.

3.5 Handover

3.5.1 Each driver authorised to drive the car during a rental must be over 25 and must have held a driver's licence for at least one year.

3.5.2 The credit card used to guarantee a rental must belong to one of the authorised drivers, and this driver must sign the rental contract. Other drivers must sign an "additional drivers' authorisation" form.

3.5.3 The driver who signs the rental agreement must not currently have a EU-Rent car on rental.

3.5.4 Before releasing the car, a credit reservation equivalent to the estimated rental cost must be made against the guaranteeing credit card.

3.5.5 The car must not be handed over to a driver who appears to be under the influence of alcohol or drugs.

3.5.6 The driver must be physically able to drive the car safely – must not be too tall, too short or too fat; if disabled, must be able to operate the controls.

3.5.7 The car must have been prepared — cleaned, full tank of fuel, oil and water topped up, tires properly inflated.

3.5.8 The car must have been checked for roadworthiness — tire tread depth, brake pedal and handbrake lever, lights, exhaust leaks, windscreen wipers.

3.6 No-shows

3.6.1 If an assigned car has not been picked up 90 minutes after the scheduled pickup time, it may be released for walk-in rental, unless the rental has been guaranteed by credit card.

3.6.2 If a rental has been guaranteed by credit card and the car has not been picked up by the end of the scheduled pickup day, one day's rental is charged to the credit card and the car is released for use the following day.

3.7 Return from rental

3.7.1 At the end of a rental, the customer may pay by cash, or by a credit card other than the one used to guarantee the rental.

3.7.2 If a car is returned to a location other than the agreed drop-off branch, a drop-off penalty is charged.

3.7.3 The car must be checked for wear (brakes, lights, tires, exhaust, wipers etc.) and damage, and repairs scheduled if necessary.

3.7.4 If the car has been damaged during the rental and the customer is liable, the customer's credit card company must be notified of a pending charge.

3.8 Early returns

3.8.1 If a car is returned early, the rental charge is calculated at the rate appropriate to the actual period of rental (e.g., daily rate rather than weekly).

3.9 Late returns

3.9.1 If the car is returned late, an hourly charge is made up to six hours' delay; after 6 hours a whole day is charged.

3.9.2 A customer may request a rental extension by phone – the extension should be granted unless the car is scheduled for maintenance.

3.9.3 If a car is not returned from rental by the end of the scheduled drop-off day and the customer has not arranged an extension, the customer should be contacted.

3.9.4 If a car is three days overdue and the customer has not arranged an extension, insurance cover lapses and the police must be informed.

3.10 Car maintenance and repairs

3.10.1 Each car must be serviced every three months or 10 000 kilometres, whichever occurs first.

3.10.2 If there is a shortage of cars for rental, routine maintenance may be delayed by up to 10% of the time or distance interval (whichever was the basis for scheduling maintenance) to meet rental demand.

3.10.3 Cars needing repairs (other than minor body scratches and dents) must not be used for rentals.

3.11 Car purchase and sale

3.11.1 Only cars on the authorised list can be purchased.

3.11.2 Cars are to be sold when they reach one year old or 40 000 kilometers, whichever occurs first.

3.12 Car ownership

3.12.1 A branch cannot refuse to accept a drop-off of a EU-Rent car, even if a one-way rental has not been authorised.

3.12.2 When a car is dropped off at a branch other than the pick-up branch, the car's ownership (and, hence, responsibility for it) switches to the drop-off branch when the car is dropped off.

3.12.3 When a transfer of a car is arranged between branches, the car's ownership switches to the "receiving" branch when the car is picked up.

3.12.4 In each car group, if a branch accumulates cars to take it more than 10% over its quota, it must reduce the number back to within 10% of quota by transferring cars to other branches or selling some cars.

3.12.4 In each car group, if a branch loses cars to take it more than 10% below its quota, it must increase the number back to within 10% of quota by transferring cars from other branches or buying some cars.

8.3 Demonstration and evaluation per perspective

8.3.1 Perspective 1: Business rules

8.3.1.1 Terms

Terms are descriptions of words or phrases with specific meaning to the organisation (section 3.3). The proposed technique can represent terms as follows:

Example 1 – Term

Original source statement

2.2.1 Most rentals are by advance reservation; the rental period and the car group are specified at the time of reservation.

Modelled statements

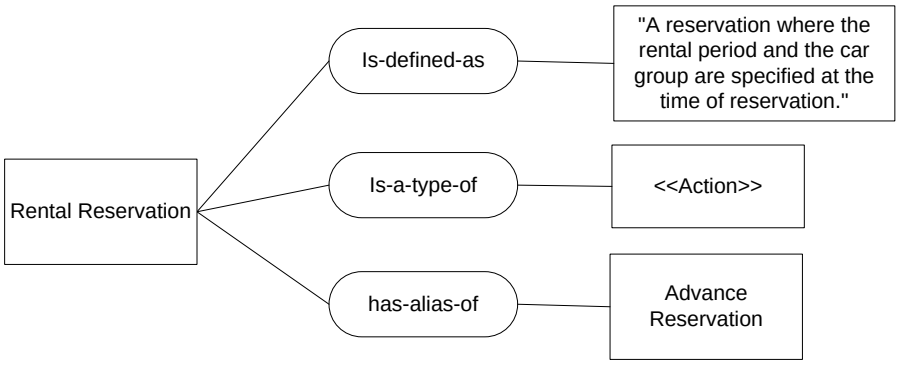
No	Ref	Link	Subject	Predicate	Operator	Qualification	Object - Complement
			Advance Reservation	is-defined-as			"A reservation where the rental period and the car group are specified at the time of reservation."
			Advance Reservation	is-a-type-of			<<Action>>

Two statements are needed to define any term. The first describes its definition and the second its type in terms of the modelling entities described in the previous chapter.

Optional terms can be added to describe aliases as needed, for instance (not in original case study, just for illustration):

No	Ref	Link	Subject	Predicate	Operator	Qualification	Object - Complement
			Pre-reservation	is-alias-of			Advance Reservation

Graphical representation



Example 2 – Term

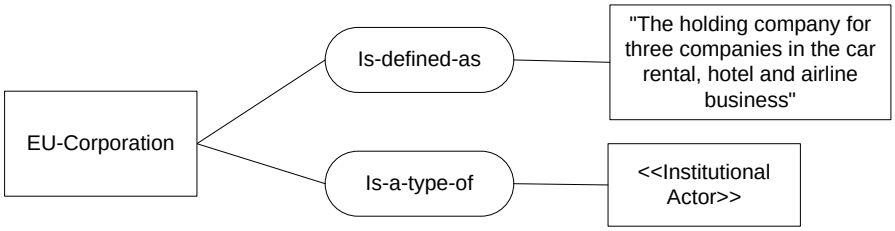
Original source statement

1.1.1 EU-Rent is a car rental company owned by EU-Corporation. It is one of three businesses – the other two being hotels and an airline – that each has its own business and IT systems, but with a shared customer base.

Modelled statement

No	Ref	Link	Subject	Predicate	Operator	Qualification	Object - Complement
			EU-Corporation	is-defined-as			"The holding company for three companies in the car rental, hotel and airline business"
			EU-Corporation	is-a-type-of			<<Institutional Actor>>

Graphical representation



8.3.1.2 Facts

Facts are the relationships between two or more terms (section 3.3). The proposed technique can represent facts as follows:

Example 3 – Facts

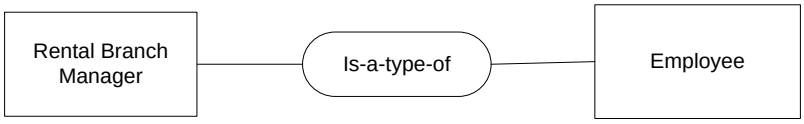
Original source statement

2.2.2 ... the branch manager may ask other branches if they have cars they can transfer to him/her. (Implied from this: *Rental branch manager is a type of employee.)*

Modelled statement

No	Ref	Link	Subject	Predicate	Operator	Qualification	Object - Complement
			Rental Branch Manager	is-a-type-of			Employee

Graphical representation



Example 4 – Facts

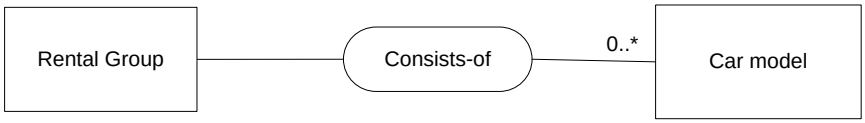
Original source statement

*3.2.1 If a rental request does not specify a particular car group or model, the default is group A (the lowest-cost group). (*Implied from this: *A rental group is composed of car models.)*

Modelled statement

No	Ref	Link	Subject	Predicate	Operator	Qualification	Object - Complement
			Rental Group	consists-of		0..*	Car model

Graphical representation



Example 5 – Facts

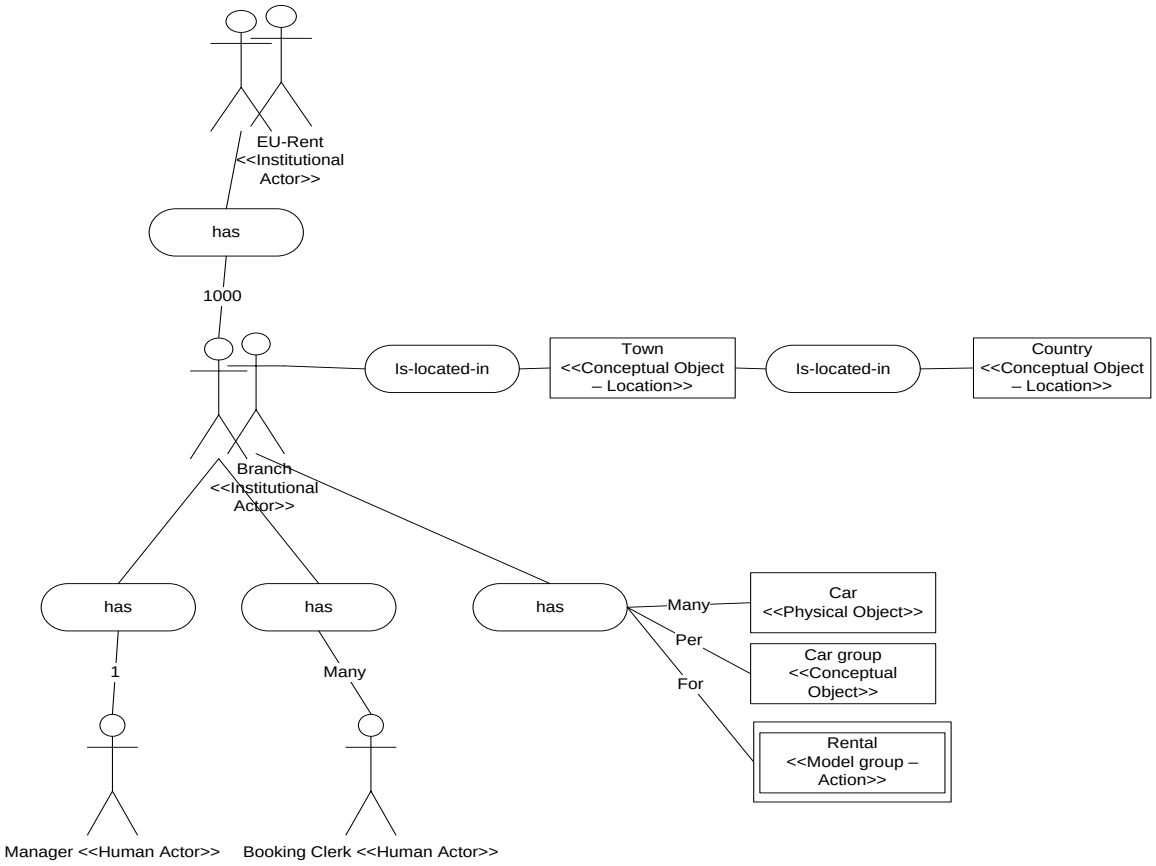
Original source statement

2.1.1 EU-Rent has 1 000 branches in towns in several countries. At each branch, cars, classified by car group, are available for rental. Each branch has a manager and booking clerks who handle rentals.

Modelled statement

No	Ref	Link	Subject	Predicate	Operator	Qualification	Object - Complement
			EU-Rent	has		1000	Branch
			Branch	is-a-type-of			<<Institutional Actor>>
			Branch	is-located-in			Town
			Town	is-located-in			Country
			Town	is-a-type-of			<<Conceptual Object - Location>>
			Country	is-a-type-of			<<Conceptual Object - Location>>
			Branch	has		Many	Car
					Per		Car Group
					For		Rental
			Car	is-a-type-of			<<Physical Object>>
			Car Group	is-a-type-of			<<Conceptual Object - List>>
			Branch	has		1	Manager
			Branch	has		Many	Booking Clerk

Graphical representation



Example 6 – Facts

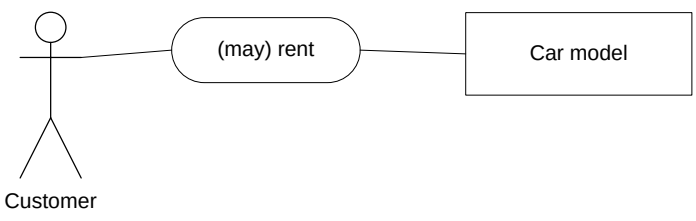
Original source statement

Car models may be requested by customers.

Modelled statement

No	Ref	Link	Subject	Predicate	Operator	Qualification	Object - Complement
			Customer	May Request			Car model

Graphical representation



Example 7 – Facts

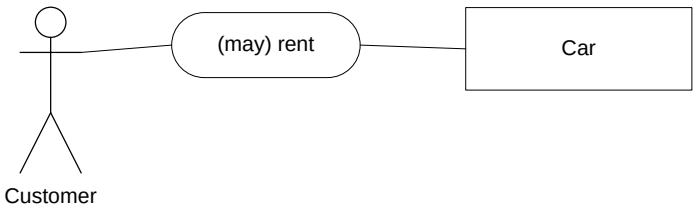
Original source statement

Cars may be rented by customers.

Modelled statement

No	Ref	Link	Subject	Predicate	Operator	Qualification	Object - Complement
			Customer	(May) Rent			Car

Graphical representation



8.3.1.3 Constraints or action assertions

Example 8 – Constraint

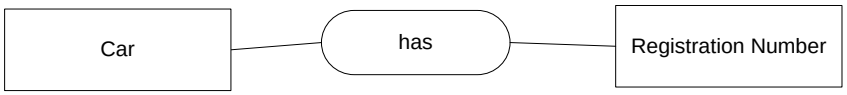
Original source statement

A car must have a registration number.

Modelled statement

No	Ref	Link	Subject	Predicate	Operator	Qualification	Object - Complement
			Car	Has			Registration Number

Graphical representation



Example 9 – Constraint

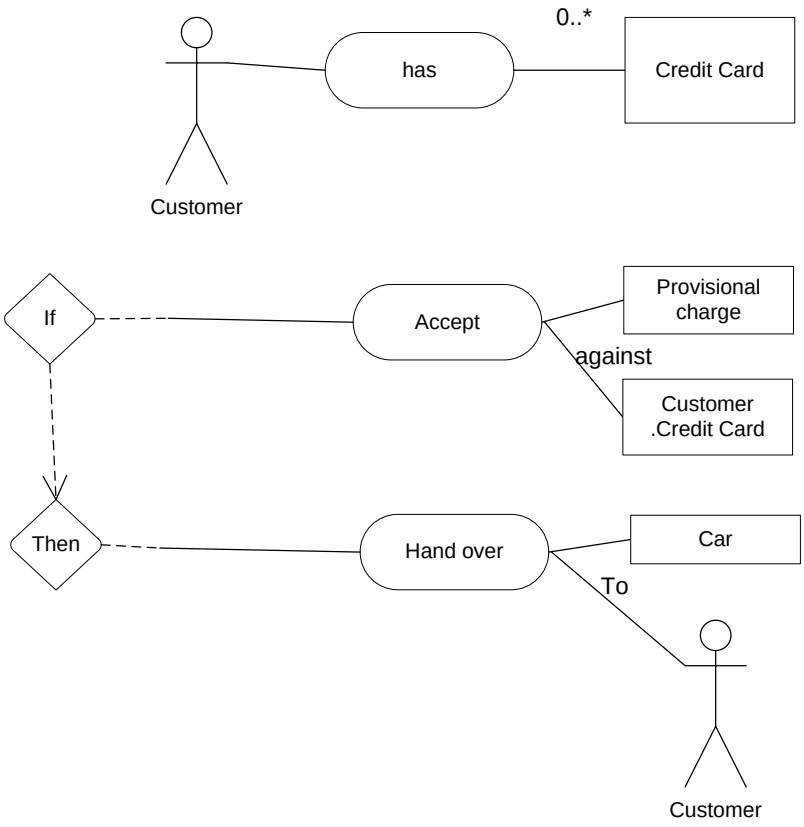
Original source statement

A car cannot be handed over to the customer unless a provisional charge has been accepted against the customer's credit card.

Modelled statement

No	Ref	Link	Subject	Predicate	Operator	Qualification	Object - Complement
			Customer	has		0..*	Credit Card
		If	<<null>>	Accept			Provisional charge
					Against		Customer.Credit Card
		Then	<<null>>	Hand-over			Car
					To		Customer

Graphical representation



Example 10 – Constraint

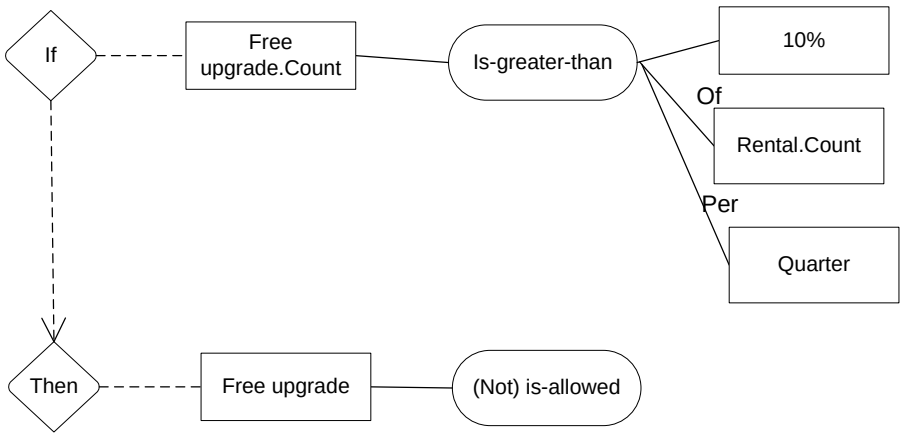
Original source statement

In each quarter, no more than 10% of rentals should have free upgrades.

Modelled statement

No	Ref	Link	Subject	Predicate	Operator	Qualification	Object - Complement
		If	Free upgrade.Count	is-greater-than			10%
						Of	Rental.Count
						Per	Quarter
		Then	Free upgrade	(Not) is-allowed			

Graphical representation



8.3.1.4 Derivations

Example 11 – Derivation

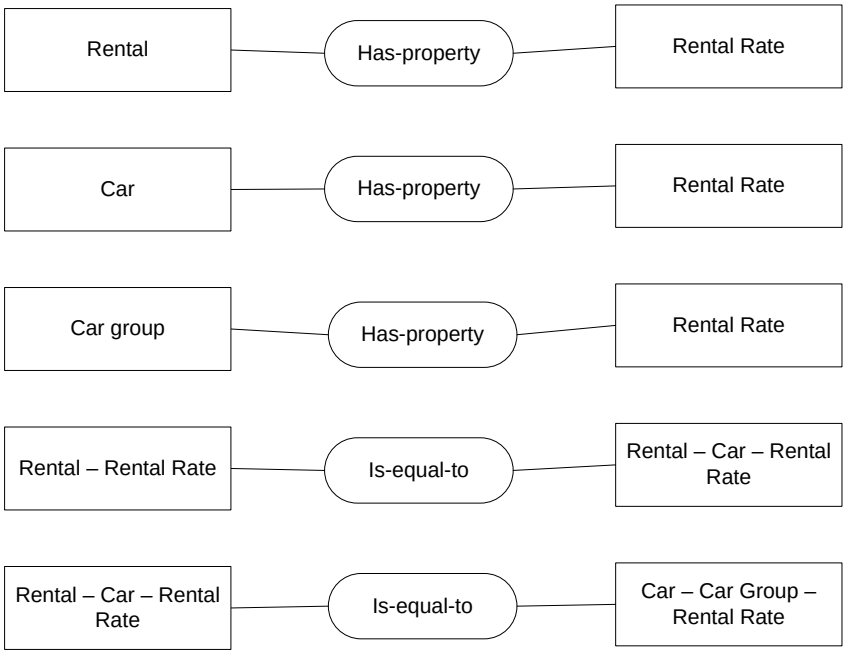
Original source statement

The "rental rate" in RENTAL is inferred from the "rental rate" of the CAR of that RENTAL, through a many-to-one relationship. This, in turn, is inferred from the "rental rate" of the CAR GROUP that the CAR is in.

Modelled statement

No	Ref	Link	Subject	Predicate	Operator	Qualification	Object - Complement
			Rental Rate	is-property-of			Rental
			Rental Rate	is-property-of			Car
			Rental Rate	is-property-of			Car Group
			Rental.Rental Rate	is-equal-to			Rental.Car.Rental Rate
			Rental.Car.Rental Rate	is-equal-to			Car.Car Group.Rental Rate

Graphical representation



Example 12 – Derivation

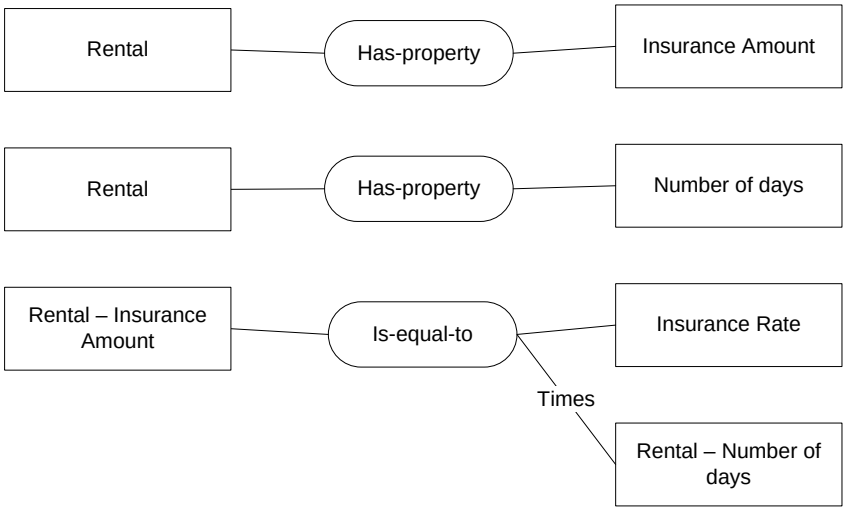
Original source statement

The "insurance amount" in RENTAL is calculated from the "insurance rate" multiplied by the "number of days".

Modelled statement

No	Ref	Link	Subject	Predicate	Operator	Qualification	Object - Complement
			Insurance Amount	is-property-of			Rental
			Number of days	is-property-of			Rental
			Rental.Insurance Amount	is-equal-to			Insurance Rate
					times		Rental.Number of days

Graphical representation



Example 13 – Derivation

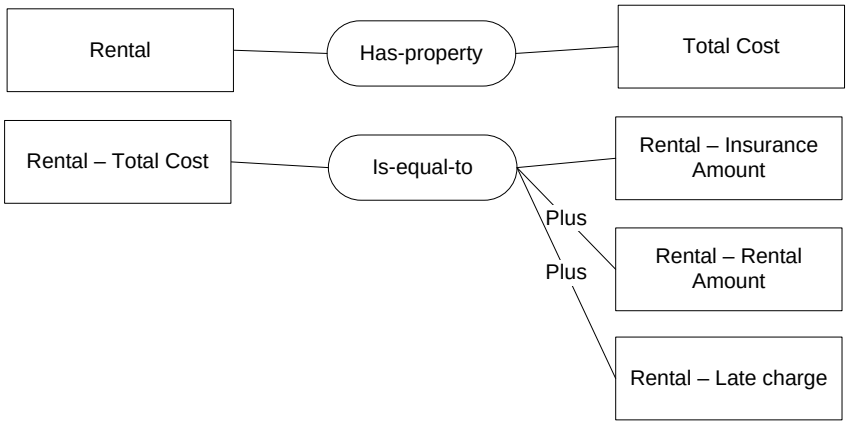
Original source statement

The "total cost" of the RENTAL is calculated from the sum of "insurance amount", "rental amount" and "late charge".

Modelled statement

No	Ref	Link	Subject	Predicate	Operator	Qualification	Object - Complement
			Total Cost	is-property-of			Rental
			Rental.Total Cost	is-equal-to			Rental.Insurance Amount
					plus		Rental.Rental Amount
					plus		Rental.Late charge

Graphical representation



Example 14 – Derivation

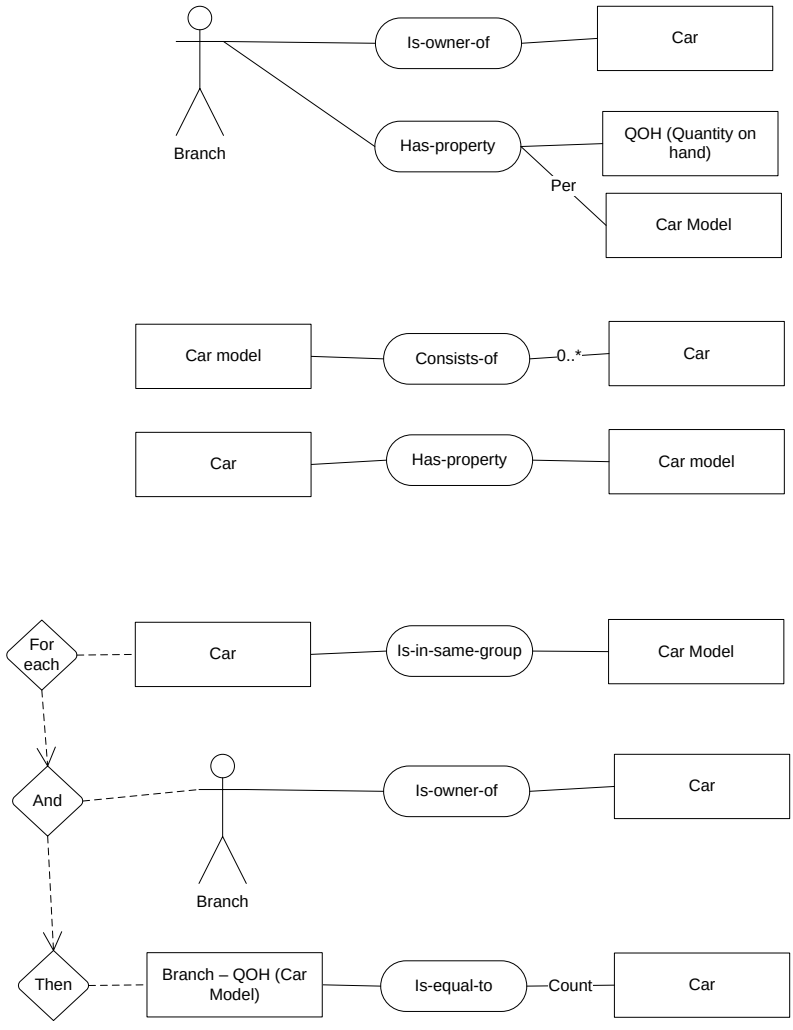
Original source statement

The branch inventory of a car model is composed of the cars of that model owned by the branch.

Modelled statement

No	Ref	Link	Subject	Predicate	Operator	Qualification	Object - Complement
			Branch	is-owner-of		0..*	Car
			Car model	consists-of		0..*	Car
			Car	has-property			Car model
			Branch	has-property			QOH (Quantity on hand)
					Per		Car model
		ForEach	Car	is-in-same-group			Car model
		And	Branch	is-owner-of			Car
		Then	Branch.QOH(Car Model)	is-equal-to		Count	Car

Graphical representation



8.3.2 Perspective 2: ISD modelling

The integrative technique explained in this section can be introduced fairly easily to non-technical users, because it is in essence just a slightly more formalised extension of natural language. For instance, instead of allowing passive sentences, only active sentences are permitted, sentences must always follow the basic subject-predicate-object/complement form, and only predefined (by the users themselves) words can be used.

The analysis of EU-Rent using this modelling technique is used to illustrate the application of the technique to a typical ISD analysis. The application is illustrative and not exhaustive. Two main phases can be identified: a business modelling phase involving the business user and a systems modelling phase translating the business model into existing ISD modelling structures.

Phase 1: Business modelling

Step 1: Identify base entities

1.1 Question to users to determine actors: Give a list of all the human individuals, institutions and intelligent actors (like IS and smart devices) both inside and outside your organisation with which the proposed system will interface.

Actors:	**Institutional** EU-Rent, car rental company, EU-Corporation, business, hotel, airline, EU-Fly, EU-Stay, branch, renting branch, return branch, pickup branch **Individual** Branch manager, clerk, customer, driver **Artificial** IT systems (1.1.1)

1.2 Question to users to determine objects: Give a list of physical objects (things that you can handle and touch), informational objects (objects that store, manipulate, input or display information on any media such as paper, spreadsheets or MSWord documents) and conceptual objects (any classification systems, anything that represents a place and any time-related aspect).

Objects:	**Physical** Car **Conceptual** Town (place), country (place), car group, rental period (time), bad experience type (e.g. late return, problems with payment or damage to cars), pickup day (time). **Informational** Rental, transfer, advance reservation, immediate rental, bad experience.

1.3 At this stage no questions are asked about acts/relations. They can be determined as model sentences and phrases are developed.

Act/relations:	None for now.

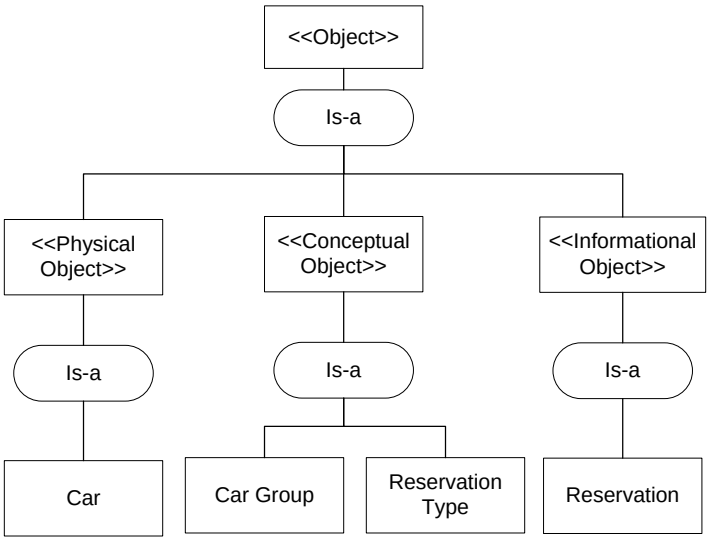
Step 2: Identify structure and role entities

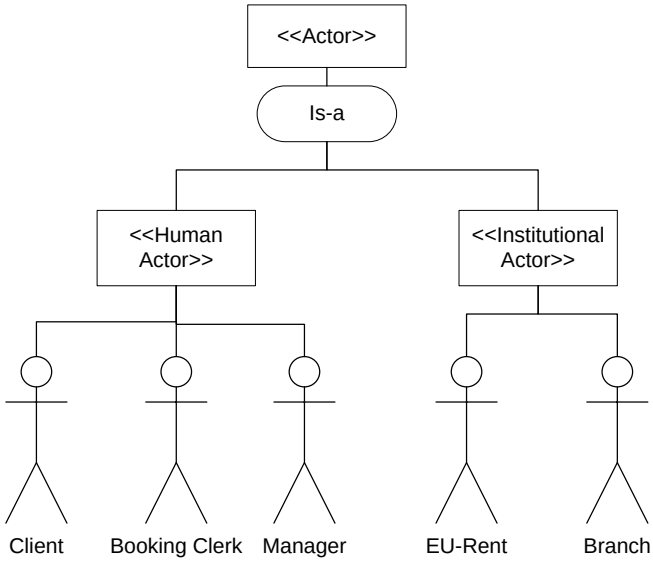
Note that it is not really possible during analysis to do structure and role entities separately. As you do structure entities, you will do them using the role entities as a logical grouping to aid analysis.

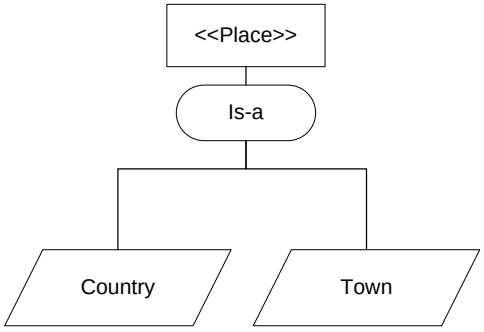
2.1 Question to users to determine models: Divide the proposed system into logical parts (according to any classification – there is no right or wrong).

Models:	Customer interface model, branch management model.

2.2 At this stage, no questions will be asked about model views. But during any stage specific views can be derived of the model to highlight just one aspect. The following three diagrams illustrate possible object, actor and place views.







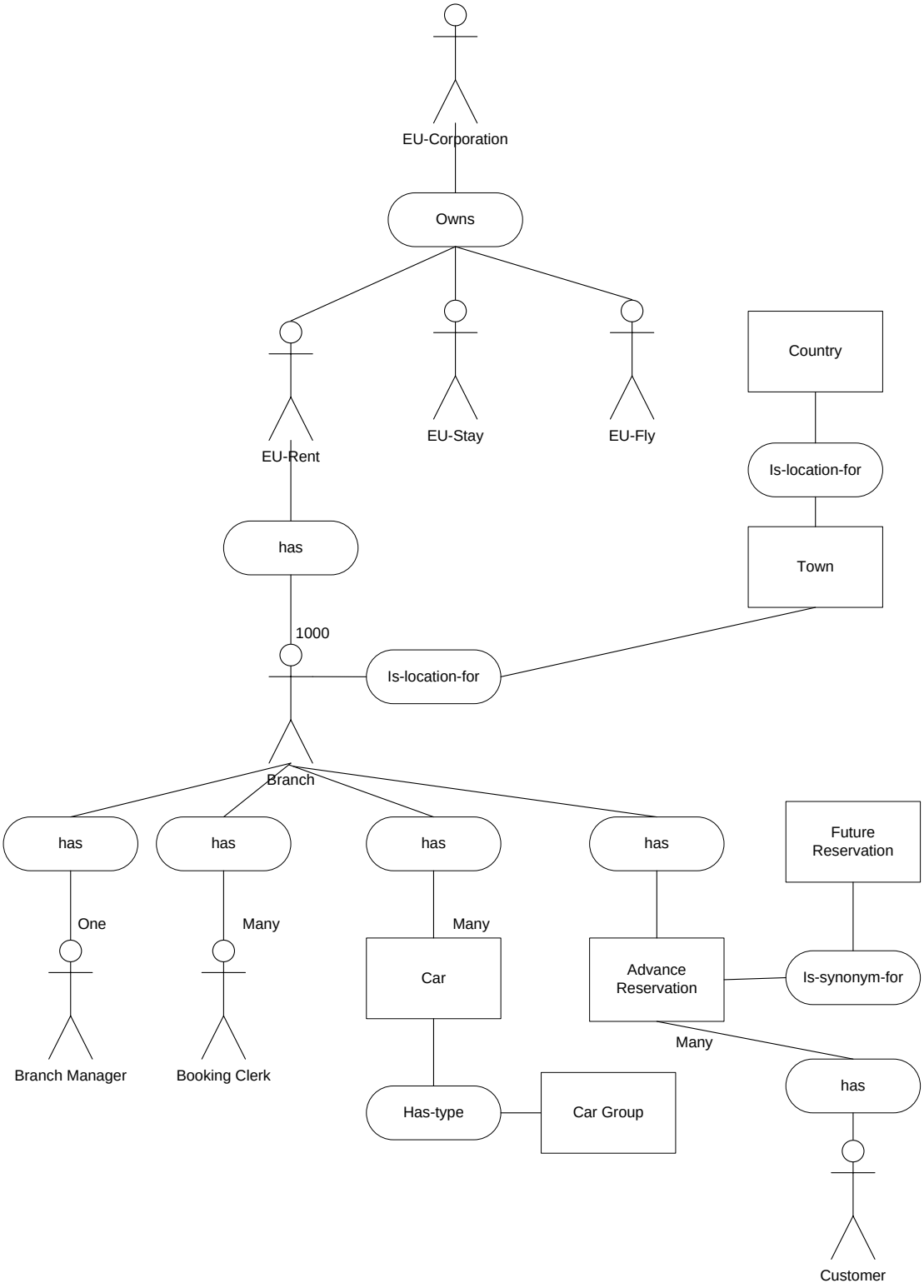
2.3 Questions to users to determine model sentences and phrases (per model):

2.3.1 What are the static relationships between things (both actors and objects)?

The easiest way is to handle each type of relationship separately.

Model sentences and phrases	
Relationships	**Client interface model** *Synonyms* Future reservation is-a-synonym-for advance reservation (3.2.4) *Inheritance* EU-Rent is-a-type-of car rental company (1.1.1) Renting branch is-a-type-of branch (2.3.1) Pickup branch is-a-type-of branch (2.3.1) *Aggregation* EU-Rent has 1 000 branches (2.1.1) *Association* Branch has one manager (2.1.1) Branch has many booking clerks for rentals (2.1.1) Customer has many advance reservations (3.2.4) *Location (special kind of association involving places)* Branch is-located-in town (2.1.1) Town is-located-in country (2.1.1) *Ownership (special kind of association involving actors)* EU-Rent is-owned-by EU-Corporation (can also be expressed as) EU-Corporation owns EU-Rent, EU-Fly, EU-Stay (1.1.1, 1.1.2) *Properties* Advance reservation has-properties rental period, car group (2.2.1)

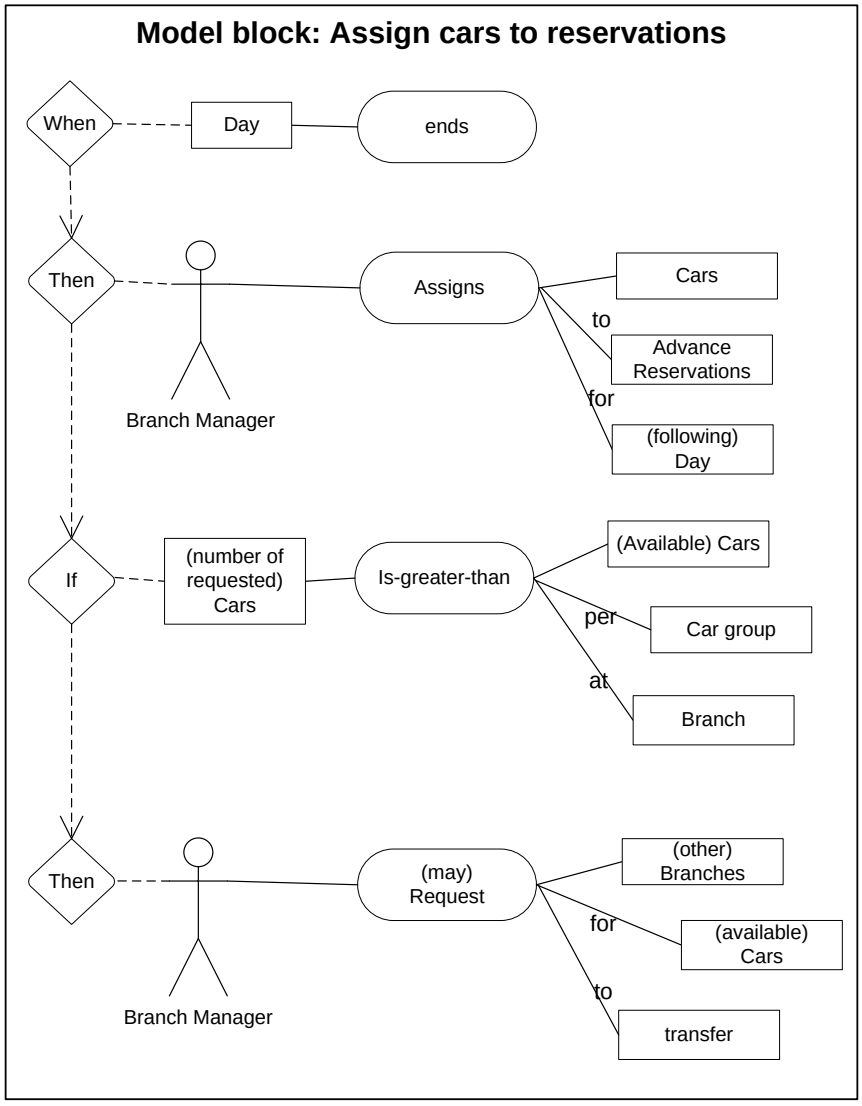
Graphical representation



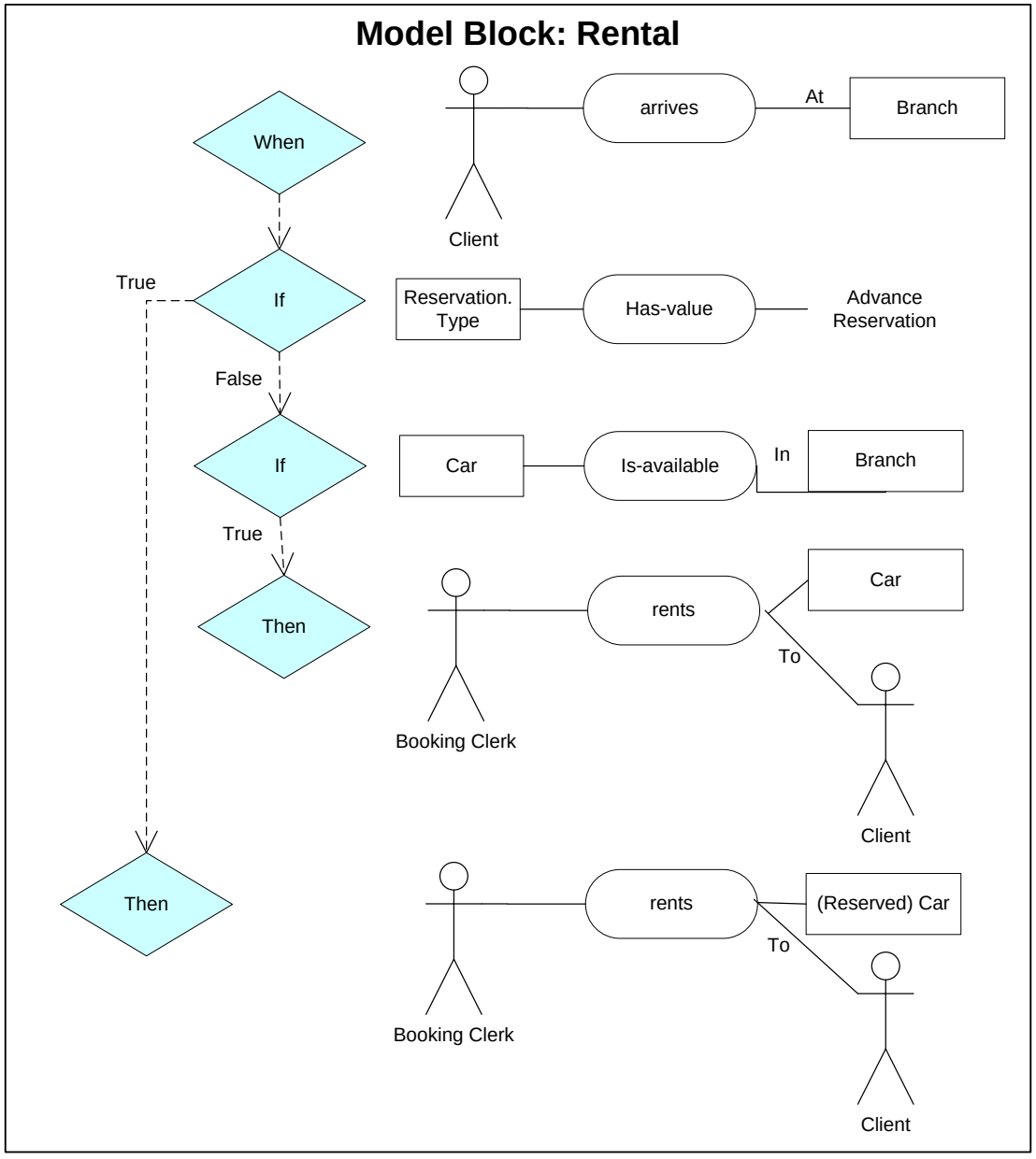
2.3.2 What are the dynamic relationships between things (both actors and objects)? Here the standard concept of a use case is one of the best ways to define the dynamic relationships between entities, where each use case is defined as a model block.

Model sentences and phrases	
Actions	**Client interface model** *Assign cars to reservations* (model block, see 2.2.2) When day ends (event) Then branch manager assigns cars to advance reservations for following day If (number of requested) cars is-greater-than available cars, per car group, at branch Then branch manager (may) request other branches for available cars to transfer

Graphical representation



If needed, more comprehensive dynamic relationships can be created to show the logical and causal relationships between model phrases. For instance, the model block for rental is as follows:



Phase 2: Systems modelling

Step 1: Create a data model

1.1 Identify entities/classes. Every*thing* (intelligent and non-intelligent object) identified is a potential entity or class.

1.2 Identify entity/class relationships. Every static relationship is a potential entity/class relationship that can be expressed by means of an ERD or class diagram (see Figure 8-1).

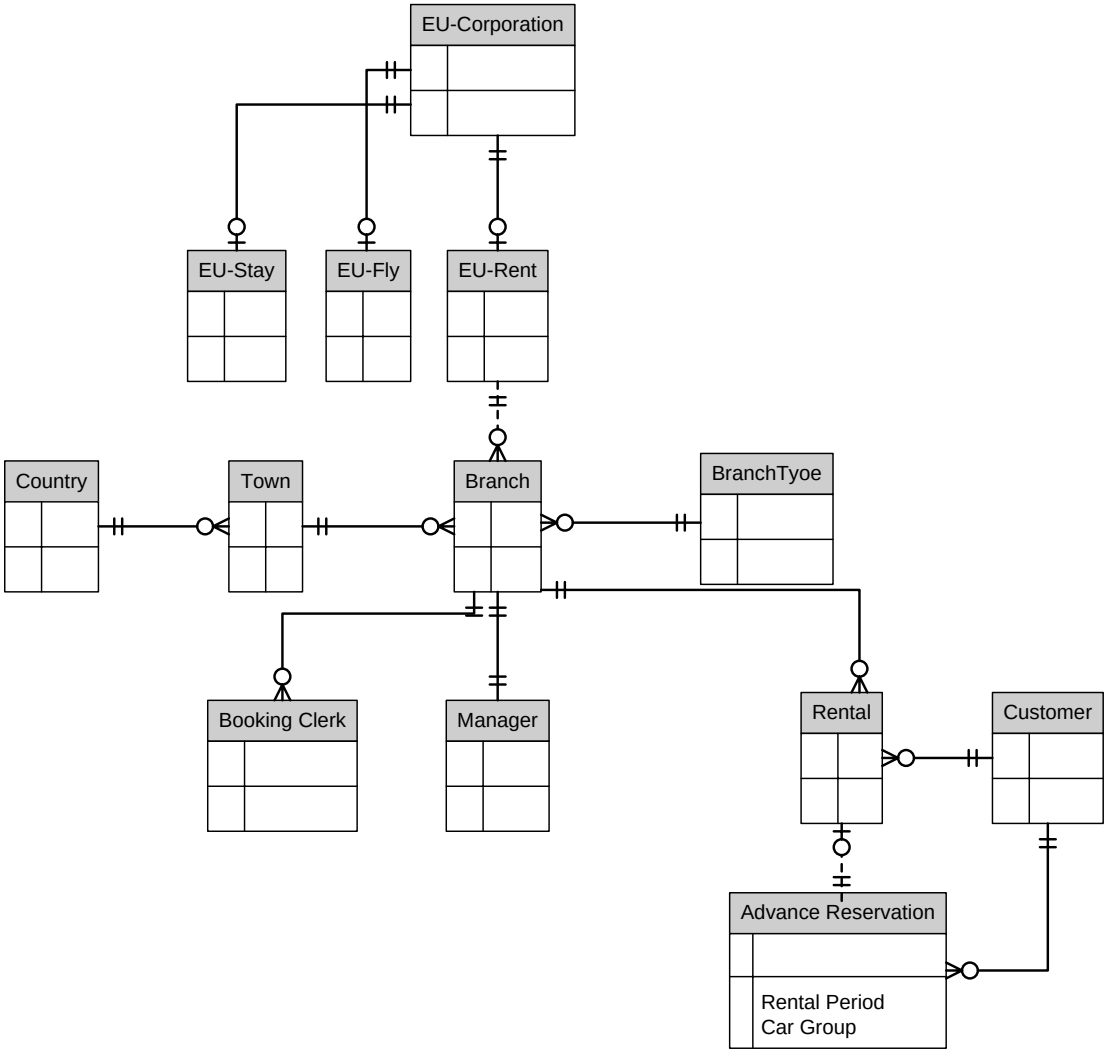


Figure 8-1: Example of class diagram developed from model

8.3.3 Perspective 3: Requirements modelling using use cases

Use case modelling has become the main formal approach (in contrast to the informal textual approach) to specifying requirements. In spite of its many benefits, a main problem is that the use case narrative remains fundamentally textual. This leads inevitably to possible ambiguity, resulting in many researchers looking for ways to overcome this problem (Yue, Briand and Labiche, 2009). This section will illustrate that the integrative modelling technique addresses this problem adequately.

An example involving use case modelling is shown to compare and evaluate it with the proposed integrative modelling technique. Diaz et al. (2004) provide an example of a use case to show an ATM withdrawal transaction (see Figure 8-2 below).

USE CASE	WITHDRAWAL
SUMMARY	IF THE CLIENT SUBSCRIBES TO THE NETWORK OF AUTOMATIC TELLER MACHINES, THE ATMS WILL ENABLE HIM TO WITHDRAW CASH FROM EXISTING FUNDS IN ANY OF THE ACCOUNTS HE MAINTAINS WITH THE BANK.
ACTOR	THE CLIENT
PRECONDITION	THE ATMS IS IN A MODE AWAITING A NEW TRANSACTION. A WELCOME MESSAGE IS SHOWN ON THE SCREEN OF THE ATM.
DESCRIPTION	**Basic Path** Withdrawal INITIATES WHEN the Client inserts a card in the slot of the ATM. 1. The ATMS reads the code on the card´s magnetic tape. 2. The ATMS verifies the validity of the card. 3. The ATMS requests that the Client introduce his password. 4. The Client introduces the digits of his personal password. 5. The ATMS verifies that the Client´s password is correct. 6. The ATMS asks the Client to select a type of transaction from a given list. 7. The Client chooses the function "withdrawal of funds". 8. The ATMS asks the Client to choose from a list the account from where the funds shall be withdrawn. 9. The ATMS asks the Client to introduce the amount of the withdrawal. 10. The ATMS conforms the request for the amount to be withdrawn with the balance in the Client´s account. 11. The ATMS dispenses the amount in bills corresponding to the amount requested. 12. The ATMS records the transaction that has taken place. 13. The ATMS updates the balance in the ATM. 14. The ATMS prints out a receipt for the Client. 15. The ATMS ejects the Client´s magnetic card. END Withdrawal. **Alternative Paths** *Card not valid or incorrect password.* 1. The ATMS informs the Client about the error. 2. The use case ends. *Amount of the withdrawal exceeding amount existing in account or amount not available at the ATM.* 1. The ATMS informs the Client about the error 2. The ATMS asks the Client to introduce another amount. 3. The ATMS executes the actions described from step 10 onwards.
POSTCONDITION	THE TRANSACTION IS RECORDED BY THE ATMS AND THE BALANCE IN THE ATMS IS UPDATED.

Diaz et al. (2004)

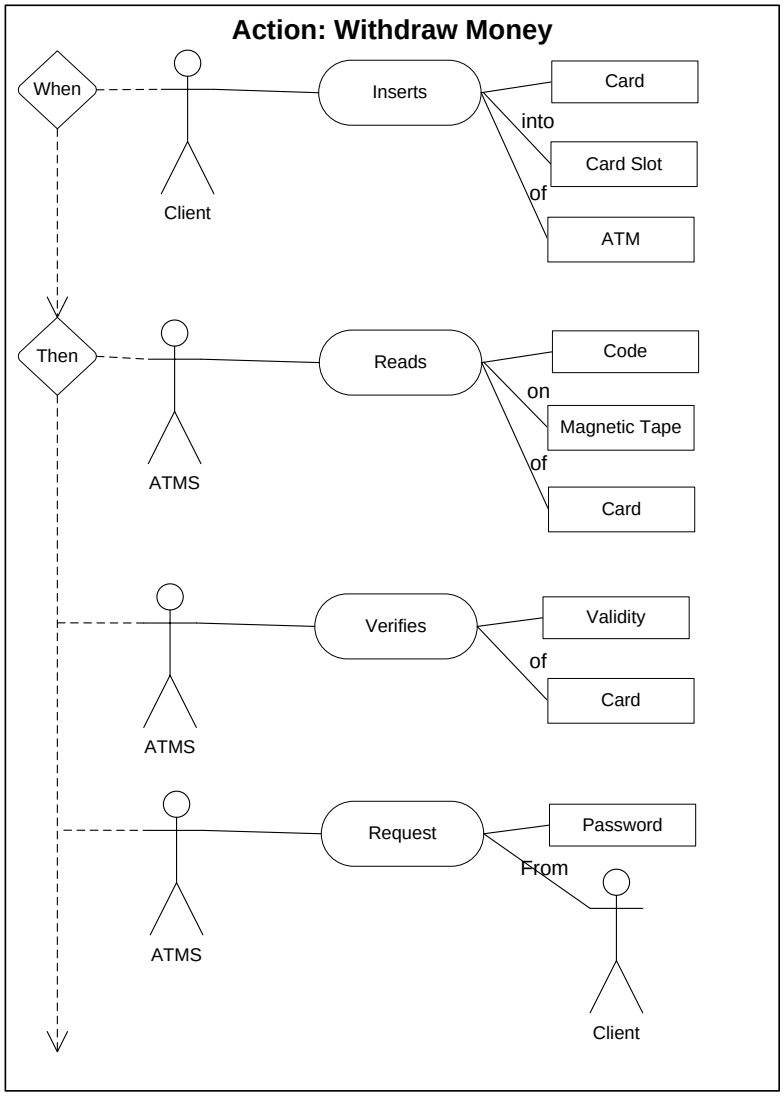
Figure 8-2: Example of use case

This basic path of the use case is partially translated into the proposed integrative modelling technique (see Table 8-1).

No	Link	Subject	Predicate	Operator	Qualification	Complement
1	When	Client	Inserts			Card
				In		Card slot
				Of		ATM
2	Then	ATMS	Reads			Code
				On		Magnetic tape
				Of		Card
3		ATMS	Verifies			Validity
				Of		Card
4		ATMS	Requests			Password
				From		Client
5		Client	Introduces			Digits
				Of	His personal	Password
6		ATMS	Verifies			Password
				Of		Client
•••						
15	End then	ATMS	Ejects			Magnetic card
				Of		Client

Table 8-1: The ATMS withdrawal use case translated into proposed technique

Graphical representation



8.3.3.1 Evaluation of use case translated into proposed technique

Before the proposed technique is evaluated, some general comments are made. This translation attempted to reproduce the use case as literally as possible. Therefore certain more correct ways of handling certain issues were not done. These are mostly related to the genitive case. For instance, the statement in line 1 "Client inserts a card in the slot of the ATM" implies that an ATM consists of a *slot* component. Therefore, more correctly the statement should have been translated to "*Client* | *Inserts* | *Card* | *In* | *ATM – Slot*", instead of the given "*Client* | *Inserts* | *Card* | *In* | *Slot* | *Of* | *ATM*". Similarly, the statement in line 6 should rather be "*ATMS* | *Verifies* | *Client – Password*".

The main advantages of the translation of the use case in the proposed technique format are as follows:

- The technique clearly distinguishes the different parts of the given statements. For instance, from the translated use steps it is clear that there are two actors (*client, ATMS*), a number of actions (e.g. *insert, read, verify, request* and *introduce*) and a number of objects involved (e.g. *card, slot, ATM, code, magnetic tape, password, digits* and *type of transaction*).
- The technique further directly and indirectly indicates the relationships between many of the objects identified by means of the **Complement operator** and **Complement qualification** columns. For instance, the preposition *of* in many cases indicates a *consists-of, has* or *ownership* relationship. For instance, statement 1 indicates that an *ATM* consists of a *slot* and statement 6 indicates that a *client* has a *password*.
- The structure of the statements forces the analyst to determine clearly what is really being communicated by the user. The natural language statements of use cases can easily be ambiguous and vague, but it is a little bit easier to be more clear and direct with the proposed technique. For instance, the distinction between the direct object of the statement and the complementary objects and their relationships are much clearer than with only natural language.

- The resultant actors, actions, relationships and objects can be translated into other modelling techniques (especially class diagrams in UML) much more directly and algorithmically than with the natural language statements. According to Liang (2003), mapping into classes can done either by (1) identifying candidate classes from the nouns and noun phrases used in use case descriptions or (2) checking to see if nouns in use case descriptions fall into the so-called classic categories of classes, such as tangible things, roles, concepts and events. The problems in identifying classes from these methods are use cases only show one of the scenarios, different words are used when describing the same thing, too many candidate classes are identified and different types of classic class categories are used by different people.

The disadvantages of the translation of the use case in the proposed technique format are as follows:

- The technique can be considered to be more complex to use by business users than the natural language statements of use cases. Some training needs to be done to explain what every column means and some of the column names can cause some users to have a mental block against its use.

8.4 Implementation of the technique as software

If the technique can be implemented by means of a software system, it will assist tremendously with the analysis of any situation. Every one of the base entities can be defined (see Figure 8-3 for an “add actor” example screen). A list of all actors is then created (see Figure 8-3).

In a fully fledged system many validations can be done and help provided. For instance, a thesaurus can warn of possible synonyms (see Figure 8-5). An ontology (see chapter 9 for an expansion of this idea) can be developed that defines common words in language with respect to this technique where, for instance, all human nouns are indicated as human actors. Normal dictionary definitions of words can be provided as a base from which to create custom definitions for terms.

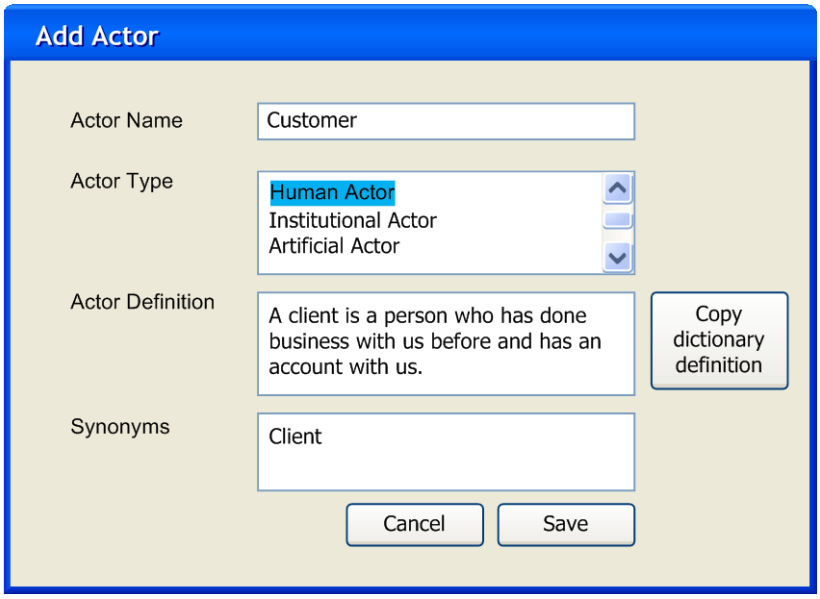


Figure 8-3: Example screen: "Add actor"

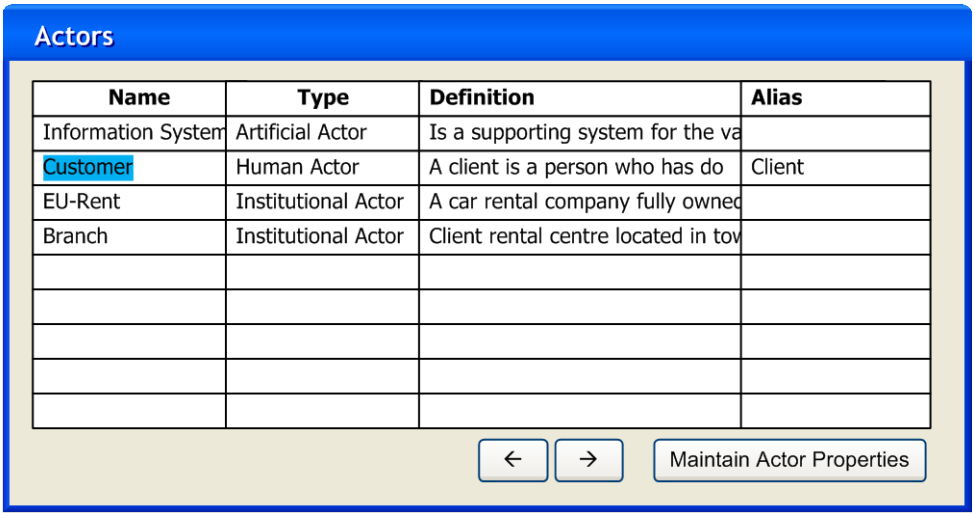


Figure 8-4: Example screen: "Display actors"

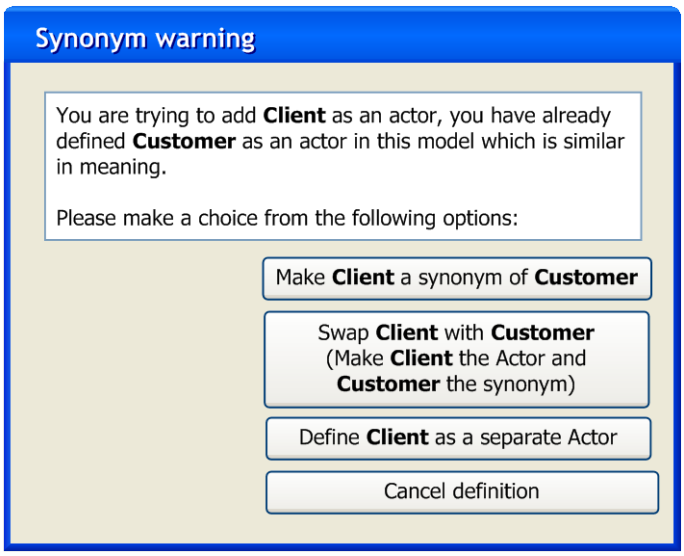


Figure 8-5: Example screen: "Synonym warning"

Model phrases and sentences can be built from the defined base entities. One will most probably define base entities as one defines model phrases. The example screen (see Figure 8-6) shows that every cell (except the **Predicate** column) has a drop-down, implying that one can only select actors from predefined actors, complements from predefined objects and so on. It is important to note that certain columns' drop-downs will have a finite domain, while others can be added to by the user. For instance, the list of **links**, **operators** and **qualifications** to choose from will be fairly finite, while **subjects** and **complements** will be specifically defined for each situation. In this example, predicates are left open-ended, but an ontology could be developed that will facilitate a finite list of possible actions that can be done by a specific kind of actor on a specific kind of object. For instance, the actions that can be done to an informational object can be limited to most probably the following: create object, delete object, rename object, add record, read record, edit record, delete record, sort records, give/remove access to object or records.

Validations can specifically be done on the relationships implied by actions. For instance, the screen in Figure 8-7 shows the typical validation that can be performed when the relationships between two objects change.

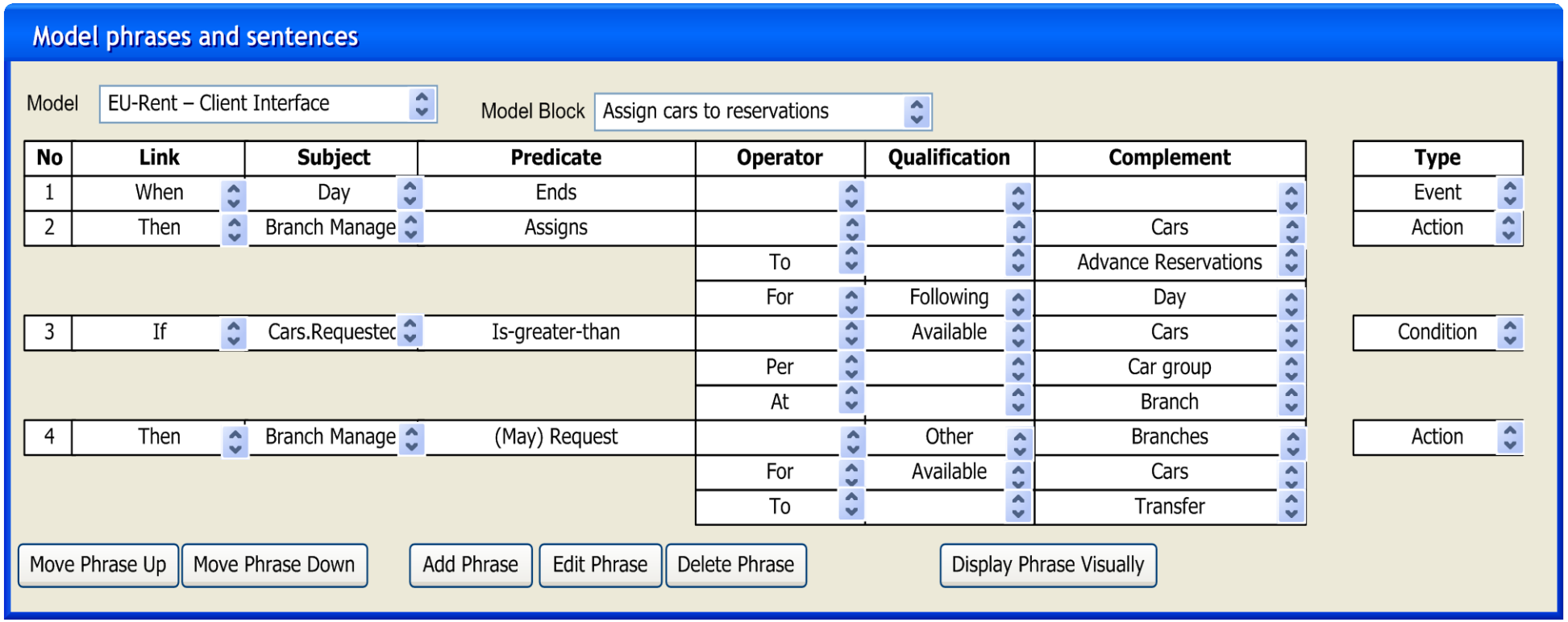


Figure 8-6: Example screen: "Model phrases and sentences"

As relationships between objects are more fully developed, the system can ensure that previous definitions are brought in line with changes. For instance, assume that the ATM system has up to a point defined “ATM” and “slot” as two separate physical objects with no relationship between them. When the user then defines a *consists-of* part-whole relationship between them, the system should then allow the user to reconsider all previous uses of the two physical objects. See Figure 8-7 for an example.

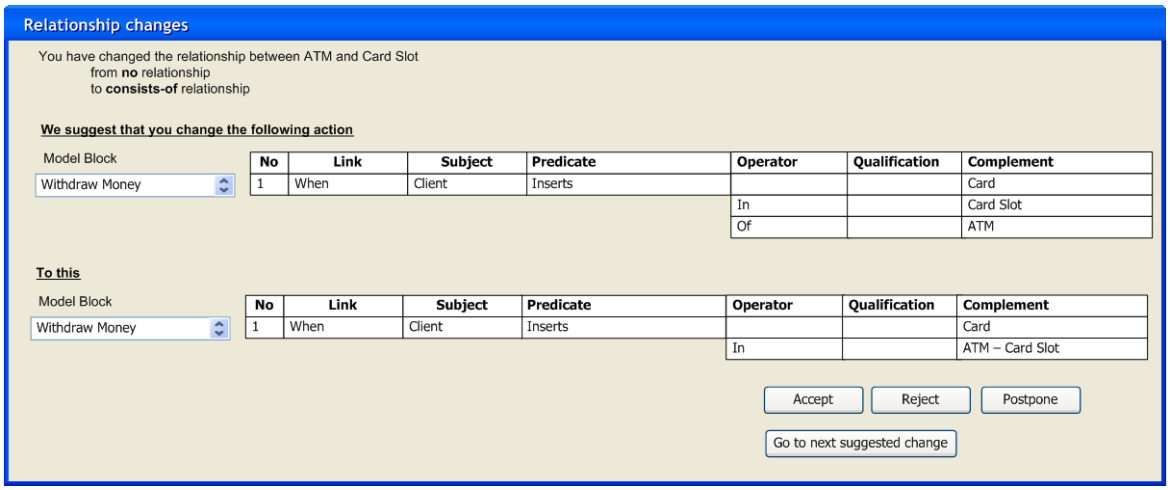


Figure 8-7: Example screen: “Relationship changes”

An information system will provide the following benefits to the analyst:

- Once a term has been defined and used once, it will be available on a drop-down list from then onwards.
- The problem of synonyms can be addressed much better. For instance, the system (using a thesaurus) can warn users when they type in “Customer” that “Client” has already been defined and that a possible synonym can be created.
- At any stage, the relationships in which any modelling entity is involved can be shown both textually and graphically to guide the analyst during the analysis, for instance, the defining and other relationships of an advance reservation (see Figure 8-9 for a visual display).

- Model views can be generated of the model based on any criteria, for instance, display all actors.

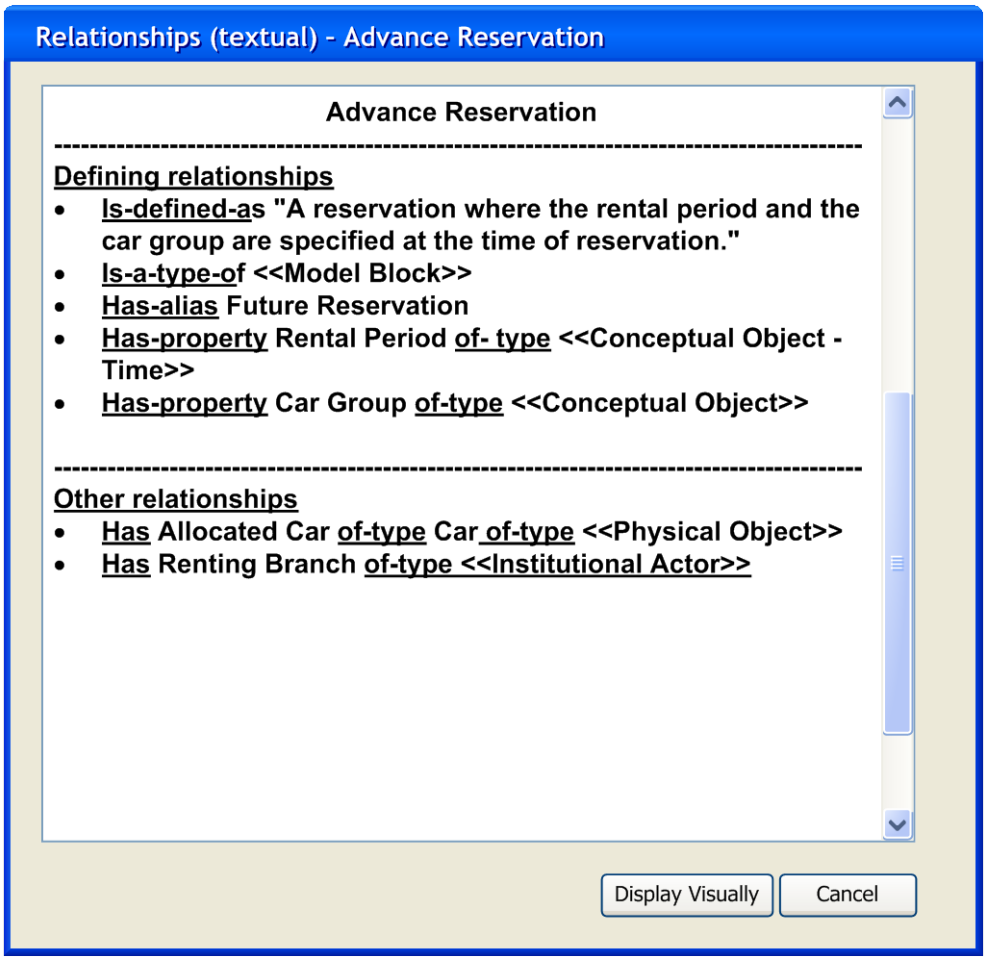


Figure 8-8: Display relationships of an object – textual format

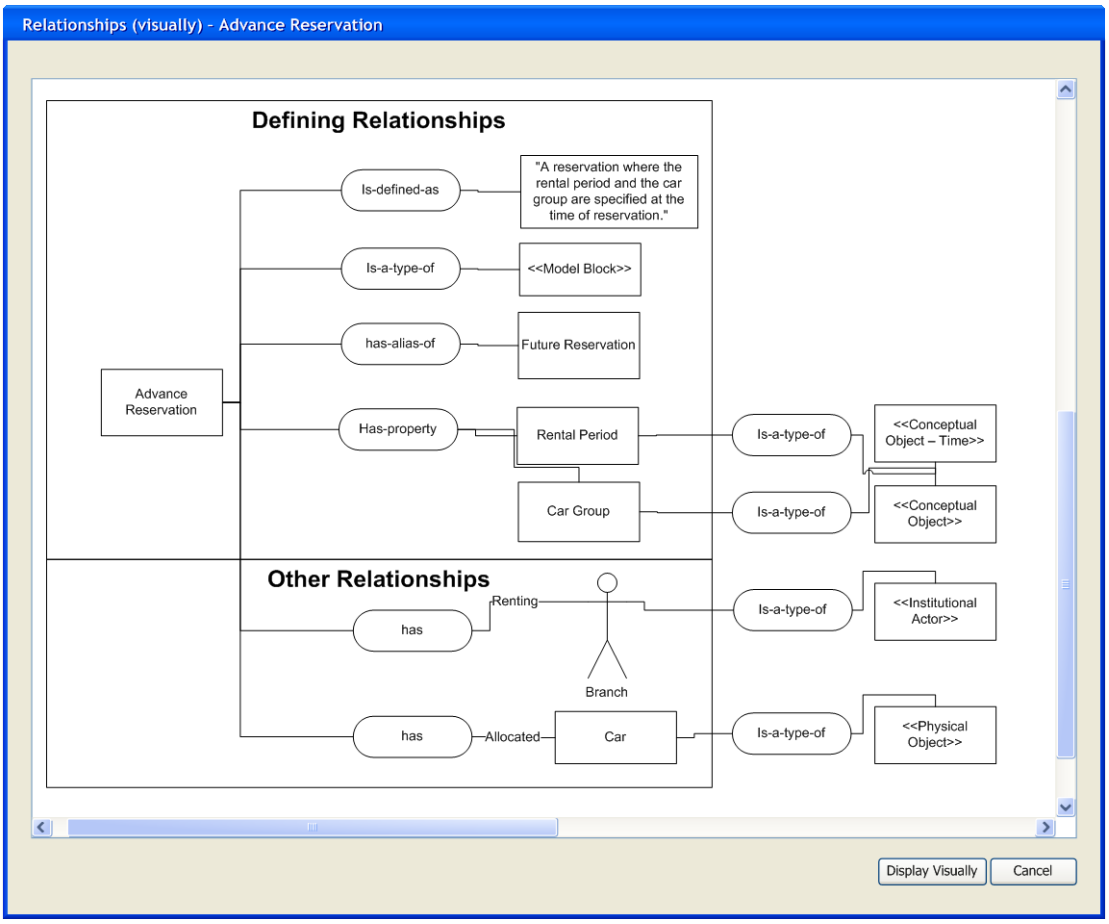


Figure 8-9: Display relationships of an object – visual format

8.5 Linking the integrative technique back to existing ISD techniques

After showing that the integrative technique can handle most of the situations involving business and ISD integration, it is also important to show that the constructs in the technique can be related to corresponding constructs in business and ISD modelling techniques. This will show that models in this technique can be translated into corresponding existing techniques.

In this section, the constructs in the ISD modelling techniques (as detailed in Chapter 7) are related to the modelling constructs of the integrative technique developed in this study (see Table 8-2).

Technique	Technique main construct	Corresponding integrative modelling technique construct
ALC/OLC	Agent	Actor
	Organisational agent	Institutional actor
	Agent life cycle	Action
	Responsibility	Relationship (between agents)
	Operations	Action
	Object	Base entity
	Object state	Base entity (state)
	Process and behavioural perspectives	Model view
AOR	Biological/human agent	Human actor
	Institutional agent	Institutional actor
	Actor actions	Action (actor)
	Actor relationships	Relationship (actor)
	Commitments, duties and rights	Relationship (between actors)
	Generalisation, composition	Relationship (type)
	Claims	Event (future)
	Internal/external agent	Actor
	Objects/entities	Object
	Entity properties/attributes	Object (properties)
	Entity type	Object (types)
	Social interaction process	Relationships (between actors)
	Non-social interaction process	Relationships (between actors and objects/artificial actors)
	Reaction rule	Action
ARM	Agency	Institutional actor
	Complex (macro) agent	Institutional actor
	Primitive agent	Human actor
	Responsibility	Relationship (between actors)
	Physical objects	Physical object, informational object, artificial actor, human actor
	Logical objects	Conceptual object
	Structural perspective	Model view
	Contractual relationships	Relationships (between actors)
	Functional and ownership relationships	Relationships (between actors and objects)
	Object	Object
	Subject	Subject
BPMN	Business entities	Institutional actor
	Participant, pool, swim lane	Actor
	Data object	Informational object
	Text or graphical information	Conceptual object
	Business process, activity, subprocess, tasks	Action, action step
	Event	Event

Technique	Technique main construct	Corresponding integrative modelling technique construct
	Event type	Event (type)
	Human level and machine level views	Model view
	Flow, gateways	Action relationships
DFD	External entity	Actor
	Data store	Informational object
	Information	Conceptual object
	Material resources	Physical object, artificial actor
	Process	Action
	Input, output	Conceptual object
	Flow	Action relationships
	Types of models: current physical, current logical, required logical, required physical	Model view
	Context	Model view
Gantt and PERT	Resource	Base entity
	Project	Action
	Perspective, view	Model view
	Work breakdown structure	Action relationships
	Predecessors	Action relationships
	Mechanism	Actor
IDEF0	Nouns or noun phrases	Base entity
	Software	Artificial actor
	Equipment, machines	Physical object, artificial actor
	Product	Object
	Raw material	Physical object
	Systems	Artificial object
	Function	Action
	Input, output	Object, model block
	Control	Action, object or actor
	Functional and context view	Model view
	Entity	Actor
IDEF1	Dictionary	Informational object
	Physical entity	Physical object, informational object, artificial actor, human actor
	Abstract entity	Conceptual object
	Entity class	Base entity
	Key	Object (property)
	Information view	Model view
	As-is, to-be	Model view (type)
	Relationship	Relationship
	Things	Base entity

Technique	Technique main construct	Corresponding integrative modelling technique construct
IDEF1X	Relationships	Relationships
	Synonyms, aliases, non-standard names.	Base entity (property)
	Primary key, foreign key, candidate key, alternate key	Base entity (property)
	Domain	Base entity (property)
	Verbs	Actions, relationships
	Semantic view	Model view
	Relationship	Relationship
	Object	Base entity
IDEF3	Noun or noun phrases	Base entity
	State condition types	Base entity state (types)
	Facts, constraints	Relationships, action steps
	Process, units of behaviour	Action
	Scenario	Action
	Objective view	Model view
	Link, junctions	Action relationship
	Kind and term	Any entity
IDEF5	Essential, accidental or defining properties	Any entity (property)
	Ontology, taxonomy	Conceptual object
	Vocabulary, terminology	Informational object
	Process	Action
	Relations	Relationship
	Organisation	Institutional actor
LAP	Actors	Actor
	Actor cycle	Action
	Agenda	Action
	Actor role	Action
	Authorisation/delegation/ propagation	Relationship (between actors)
	Production, coordination and communication acts	Action
	Transaction	Action
	Initiator/customer	Actor (type)
	Fact	Base entity, action step, relationship
	Event	Event
	Atomic, fibre and molecular layers	Model view
	Semiotic layers	Model view
	Action rules	Action
	Agent	Actor

Technique	Technique main construct	Corresponding integrative modelling technique construct
ODP	Agent role	Action
	Structuring rules (obligation, permission, prohibition)	Actions step or relationship
	Artefact	Object, artificial actor
	Artefact role	Action
	Service	Action
	Node	Place, artificial actor
	Enterprise, information, computational, engineering and technology perspectives/ viewpoints	Model view
	Role	Action
RAD	State	Base entity (property)
	Activities	Actions
	Trigger	Event
	Dynamics view	Model view
	Sequence	Action relationship
	Permission	Relationship
SQL	Database action	Action
	Database object, schema, table, virtual table,	Informational object
	Constraints	Action step, relationship
	Data type	Informational object (property)
	Action, transaction	Action
	Trigger	Action
	Database view	Model view
	Control-of-flow	Action relationships
	Customer/client/beneficiary/ victim/owner	Actor (type)
SSM	Actor	Actor
	Transformation	Action
	System/subsystem	Agent
	Soft systems view	Model view
	Actor	Actor
UML	Object/class	Base entity
	Attribute	Base entity (property)
	Interface	Informational object + action
	Package	Informational object
	Software component	Artificial actor (component)
	Node	Artificial actor, place
	Use case	Action

Technique	Technique main construct	Corresponding integrative modelling technique construct
	Activity, operation	Action
	Static, dynamic, functional and implementation view	Model view
	Association	Relationship
	Decision	Action
Zachman	Who	Actor
	What	Object
	How	Action
	Where	Place
	Primitives vs. composites	Relationships
	Planner, owner, designer, builder, subcontractor perspectives	Model view

Table 8-2: Comparison between existing techniques and integrative technique

8.6 Conclusion

In this chapter, the integrative technique was evaluated using the EU-Rent case study. The evaluation was done from three perspectives:

- The **business rules perspective,** which ensures that the modelling technique can model any business rule (or any other rule for that matter).
- The **ISD perspective,** which ensures that the integrative modelling technique can be used to model any business aspect of ISD through all the phases of a typical SDLC.
- The **requirements perspective,** which ensures that the most commonly used requirement modelling tool, use cases, can be modelled using the integrative technique.

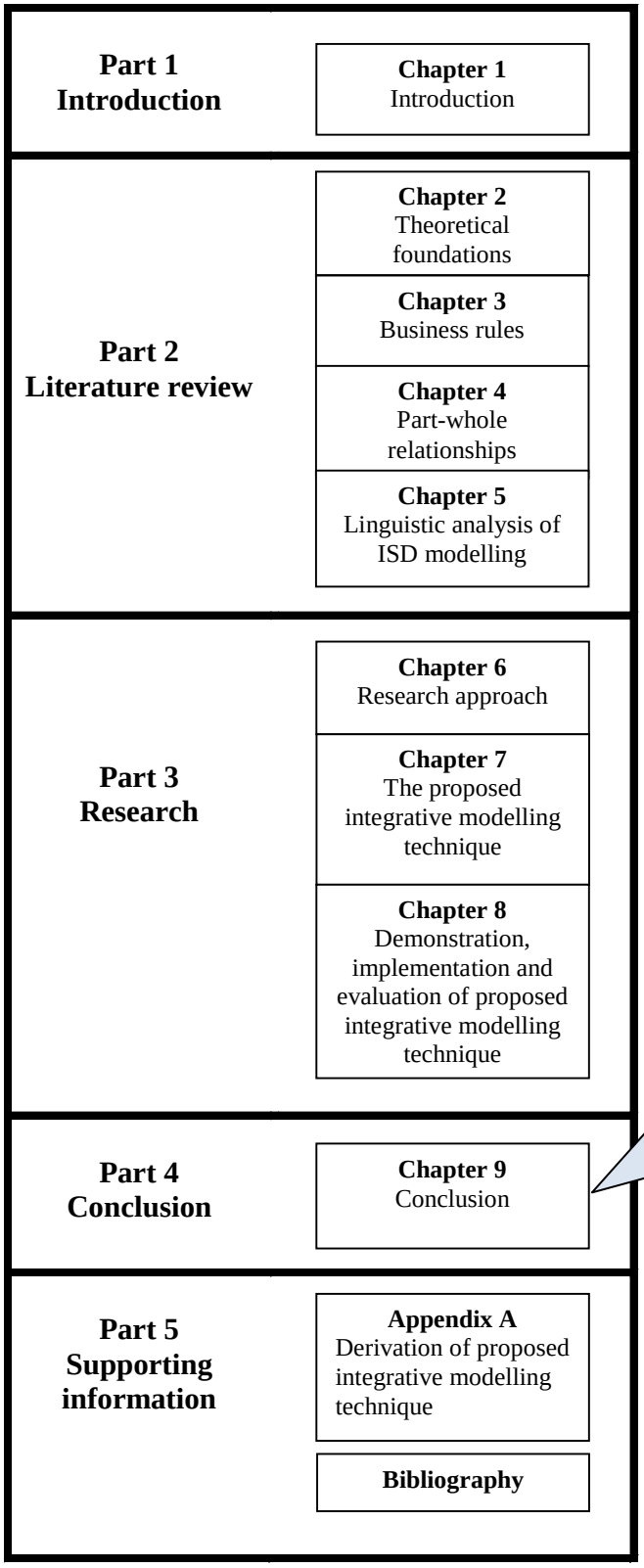
Furthermore, a software prototype was developed to illustrate how the technique can be physically instantiated. A software implementation of the technique will give many benefits to the users, mainly ensuring compliance with standards and validating the analysis.

Lastly, the integrative technique was mapped to existing ISD techniques to show that a model created with the integrative technique can be translated to existing ISD modelling techniques.

Part 4

Conclusion

9. Conclusion



9.1 Introduction
9.2 Answering the research questions
9.3 Evaluation of the research
9.4 Contribution of the research
9.5 Future research
9.6 Concluding remarks

9.1 Introduction

The goal of this study is to develop an integrative modelling technique that is easy enough to be used by most business users with little training, but robust and structured enough to be used in subsequent ISD modelling. “Integrative” refers to the fact that this technique attempts to bridge the current gap between modelling on a business level and modelling on a technical level.

The overall research methodology is design science research, using aspects of grounded theory and linguistics as the major kernel theories. These theories are used to develop the integrative technique and to base it on current ISD modelling techniques and the principles and theories of language.

In this chapter, the research questions are revisited and answered in summary. Then the research is evaluated, taking into consideration the two research approaches used, namely the grounded approach and design science research. After that, the contribution of the research is evaluated and possible further research based on this study is discussed.

9.2 Answering the research questions

The problem statement of this study is:

The current modelling techniques do not bridge the gap between business and ISD.

The main research question is:

Can an integrative modelling technique be developed to bridge the gap between business and ISD?

The underlying research questions are the following:

- *Is there a gap between business and ISD that current modelling cannot fill?*
- *What are the fundamental constructs of any integrative modelling technique between business and ISD?*

- *What are the properties and attributes of these fundamental constructs?*
- *What are the relationships between these fundamental constructs?*
- *Can it be demonstrated that the proposed technique does indeed integrate business and existing modelling techniques better than existing business modelling techniques?*

9.2.1 Is there a gap between business and ISD that current modelling cannot fill?

This question was answered in section 1.3.2, where it was shown that some of the major problems with existing business modelling are the contrasting problems of users finding formal modelling too difficult and analysts and developers finding business models too informal, leading to ambiguity and the need to do analysis again when translating models during ISD.

9.2.2 What are the fundamental constructs of any integrative modelling technique between business and ISD?

The fundamental entities of business and ISD modelling can be divided into three categories: base entities (corresponding to the morphological level in linguistics), structure entities (corresponding to the syntactical level in linguistics) and role entities (corresponding to the semantic level in linguistics) (see Figure 9-1).

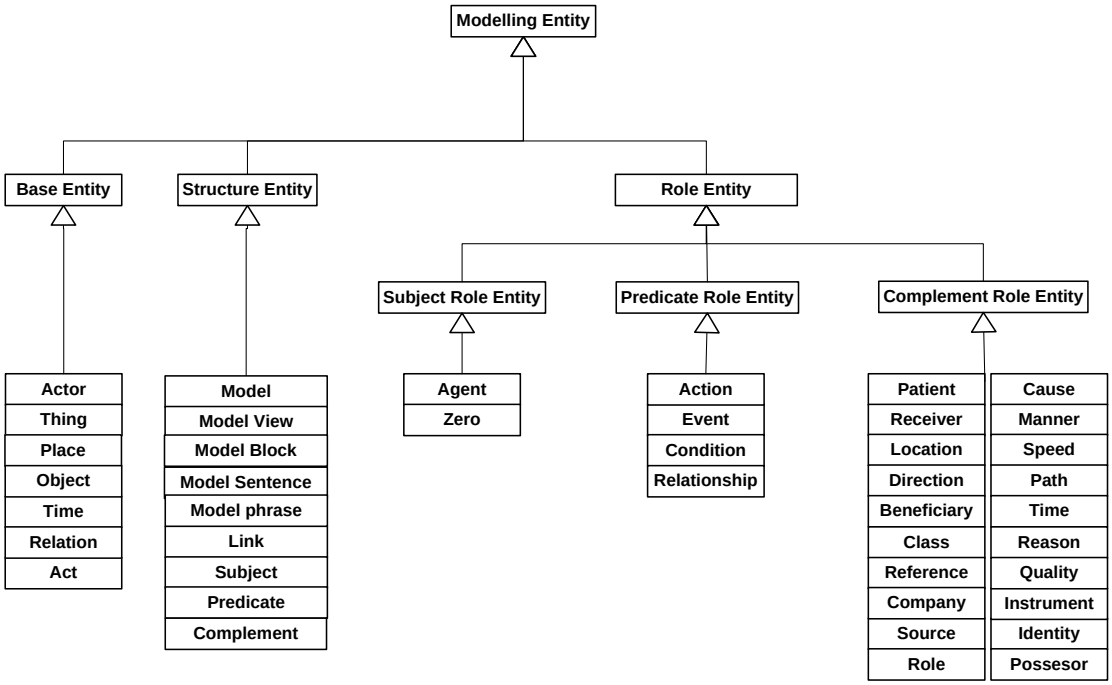


Figure 9-1: A high-level overview of the ISD modelling entities

The **base entities** are the most basic building blocks in the modelling process. They represent the real-world objects that make up organisations and systems. In the same way that verbs, nouns, adjectives, adverbs, pronouns and others make up the words of natural languages, these entities form the **words** of the proposed modelling language.

The **structure entities**, like their natural language counterparts, form the **model sentences** and **phrases** with which systems, organisations and situations can be described. They use the base entities plus specific language constructions to form these model sentences and phrases. A number of model sentences together form a **model**. Specific subsets of the model form **model views**.

The **role entities** provide insight into the meanings of the base and structure entities. For example, an actor can either play an agent role or patient role in a model phrase.

9.2.3 What are the properties and attributes of these fundamental constructs?

The main properties and attributes of these constructs can be summarised as follows:

Type	Entity	Properties and attributes	Section
Base	Actors	• Actors can autonomously perform actions and make decisions. • Actors can be classified as either human, institutional or artificial. • Human actors are primitive entities, while institutional and artificial entities are composite entities. • Actors can play various roles. • Actors have some sort of unique identification. • Actors normally have responsibility to perform certain actions. • Actors can be transformed, transported, stored, exchanged (normally their time), and controlled. • Actors can perform coordination actions like entering into commitments and complying with commitments. • Actors can perform various levels of communication actions.	7.3.1.2
	Objects	• Objects are non-autonomous, non-intelligent entities. • Objects can either be physical, informational or conceptual. • Place, time and information are specific types of conceptual objects. • Objects are composite entities. • Physical objects can have unit of measure and type of material as properties. • A place always has a type of address (physical or conceptual) and normally stores a certain kind of entity. • Objects can be transformed, transported, stored, exchanged and controlled.	7.3.1.3
	Act/relation	• An act is an active verb or verb phrase describing a dynamic action. • A relation is an auxiliary verb or verb phrase describing a static relationship.	7.3.1.4
Structure	Model	• A model represents all the information available on the situation, system or organisation under discussion.	
	Model view	• Every model can have many views of it, based on some specific aspect. • These views are subsets of the model, showing certain entities and ignoring others based on the specified criteria. • Aspects that can be considered to determine a view are type of base entity (for instance, only view actors), level of detail (for instance, only show composites and no primitive entities) and history (for example, show only the as-is, or to-	

Type	Entity	Properties and attributes	Section
		be or fifth version of a model).	
	Model block	• Model blocks are named sets of model phrases and sentences that can be referenced and used as single entities. • Model blocks have names, input and output parameters.	7.3.3.2
	Model sentences	• Model sentences are groups of model phrases that have some sort of relationship with each other. • Model sentences are formed using the *link* and *number* entities. • Typical relationships between model phrases are sequence, repetition, decision and concurrency.	7.3.3.3
	Model phrase	• Model phrases can either be events, conditions, actions or relationships (see predicate roles for more information). • Model phrases are built from *Subject*, *Predicate*, *Operator*, *Qualification* and *Complement* entities.	7.3.3.3
Role	Subject roles	• Subject roles identify the meanings that the subjects of acts/relations can have. • Subject roles are either *zero* or *agent*.	7.3.4.1
	Predicate roles	• Subject roles identify the meanings the predicate of acts/relations can have. • Predicate roles can either be actions, events, conditions or relationships. • Actions can be the transformation, transportation, storing, exchanging or controlling of entities. • Relationships between entities can either be association, property, instance, recursion, aggregation, composition, inheritance or location. • Events indicate actions that trigger other actions. • Conditions indicate relations between entities that either allow or disallow actions from taking place.	7.3.4.1
	Complement roles	• Complement roles identify the meanings that the complement of acts/relations can have. • These roles can be categorised as basic, object-related, location-related, time-related or stakeholder-related.	

9.2.4 What are the relationships between these fundamental constructs?

The relationships between these fundamental constructs are summarised below:

- The base entities **thing**, **actor** and **object** and their relationships are summarised in Figure 7-2.
- The structure entities **model**, **model view**, **model sentence** and **model phrase,** and their relationships as well as their relationships with the base entities are summarised in Figure 7-6.
- The role entities related to their corresponding base entities are summarised in Figure 7-9.

9.2.5 Can it be demonstrated that the proposed technique does indeed integrate business and existing modelling techniques better than existing business modelling techniques?

This question was answered in Chapter 8, where it was shown that from various perspectives the integrative modelling technique can be used to adequately model business aspects both easy enough for business users to use but expressive enough so that it can be translated relatively easily into existing ISD techniques.

9.3 Evaluation of the research

This research is evaluated from two perspectives. Firstly, the grounded approach that was followed to do a qualitative analysis of the data is evaluated using principles from grounded theory. Then, the research is evaluated using the criteria set out for design science research.

9.3.1 Grounded approach evaluation

It is important to understand that this study used grounded theory only as a way of doing a qualitative data analysis to inform the integrative model that was developed. It therefore does not claim to be full-scale grounded theory research. In spite of that, the grounded theory research evaluation approaches are followed to evaluate that part of the study where it is applicable.

The doubts raised by Glaser and Strauss (1967) on the applicability of the principles of rigorous quantitative research as proper criteria for judging grounded theory must be taken into account before evaluating any grounded research.

Note that in many of the evaluation questions, reference is made to existing theories. Except for a few techniques like SSM, no specific theory of ISD modelling entities exist per se, but most of the ISD modelling techniques studied have implicit theories from which these techniques were derived. When reference is made to existing theories in this section, these implicit theories embedded in the ISD modelling techniques are also implied.

Grounded theory as a specific research approach has its own set of criteria for good grounded theory research over and above the general set of criteria for qualitative research. These criteria will be addressed in this section.

Three sets of evaluations are considered:

1. Evaluation criteria from the original manuscript on grounded theory by Glaser and Strauss (1967).
2. Criteria from the one side of the current division in grounded theory, as exemplified by Strauss and Corbin (1998).
3. Criteria from the other side of the current grounded theory division, as exemplified by Glaser (1978).

9.3.1.1 Glaser and Strauss (1967)

Glaser and Strauss (1967:118) use a checklist to analyse a number of comparative studies to indicate which of these are grounded theory and which are just comparative analysis. This checklist can also be used to evaluate grounded theory, especially to determine the extent to which it goes beyond comparative analysis and actually generates theory. The checklist questions are answered one by one below.

1. Is the researcher's main emphasis upon *verifying* or *generating* theory?

This study did not consider extant theories as a primary concern, but had as primary concern the purpose of generating a set of fundamental ISD modelling entities with the purpose of creating an integrative modelling technique for further analysis.

2. Is the researcher more interested in *substantive* or *formal* theory?

Two kinds of theory can be developed: **substantive theory** considers an empirical area of enquiry, while **formal theory** considers a conceptual area of enquiry. This research concentrated on the empirical area of fundamental business and ISD modelling entities and therefore the researcher is more interested in substantive theory.

3. What is the *scope* of the theory used in the thesis?

The theory is restricted to the fundamental business and ISD entities only. During the grounded process, it became clear that the process of modelling would have been another area of possible research, but the researcher restricted the codes to only cover the fundamental entities.

4. To what degree is the theory grounded?

All of the base and structure entities came directly out of the grounded process. The detailed role entities do not directly come out of the underlying data, but emerged from the literature study done on linguistics.

5. How *dense* in conceptual detail is the theory?

The density of the conceptual detail of the theory is difficult to quantify. One way to do it is to count the number of codes and their relationships. Using that approach, there are nine base entities, 11 structure entities and 31 role entities. There are also 12 relationships between structure and base entities and 13 relationships between role and structure entities. In total, this seems to be a theory fairly dense in conceptual detail.

6. What kinds of data are used, and in what capacity, in relation to the theory?

The empirical data from the ISD techniques were used as the basis of the data, while a literature review on important concepts that emerged from the grounded analysis was used to help with the categorisation process.

7. To what degree is the theory integrated?

Every code defined can be related back to every other code. For instance, the role entity **patient** can be related to all other role codes and also all the way back to underlying structure and base entities.

8. How much clarity does the researcher reveal about the type of theory that he uses?

The researcher did not use any existing theory per se.

9.3.1.2 Strauss and Corbin (1998)

Ensuring that the central category and related concepts are integrated

According to Strauss and Corbin (1998), writing the storyline of the grounded theory will show to what extent the central category integrates all the related concepts. The storyline for this study is as follows:

> *ISD modelling is similar to linguistics. An ISD **model** can be seen as a piece of "text" (either textual or graphical) that describes a specific organisation, information system or situation in a specific modelling language. It consists of words and sentences and must comply with a set of linguistic rules defining what is allowed and what is not. The clearer and better the language and its rules are defined, the better the possibility of good communication between the various users of the modelling language.*

*The words with which ISD models are created (**base entities**) are **actors** and **objects**. These words are combined in **model phrases**. **Model phrases** form **model sentences,** which in turn can be grouped together to form submodels and **models**. These sentences and groups of sentences form **structure entities**.*

*Model phrases can be **actions, relationships, events** or **conditions.** Model phrases can be described by specifying a **subject**, **predicate, complement operator, complement qualifier** and **complement** with optional **link** to form sentences. Relationships define the structure of base and structure entities.*

*The same word can have different meanings, depending on how it is used in a sentence and the wider context. Conversely, the meaning of a specific word can influence the meaning of the wider context. Role entities can be related to the **subject**, **predicate** and **complement** parts of model phrases. Role entities can also be grouped as basic (e.g. **patient** and **instrument**), location-related (e.g. **location**, **direction** and **route**), time-related (e.g. **speed**, **frequency** and **duration**) or stakeholder-related (e.g. **beneficiary** and **company**).*

9.3.1.3 Glaser (1978)

In the original work on grounded theory, Glaser and Strauss (1967) considered three criteria with which a theory should comply: fit, relevance and the fact that it must work. Glaser (1978) adds to these a fourth criterion, modifiability, and also expands some of the other criteria.

1. Fit

Fit means that the categories of the theory (in this study embodied in the integrative technique) must fit the data and should not be forced to fit in with pre-conceived ideas or existing theories. Glaser (1978:4) is quite adamant about it: “Our position is that the reality produced in research is more accurate than the theory whose categories do not fit, not the reverse.”

Glaser further expands this category with two more properties: “refit” and “emergent fit”. The refit of categories to existing data is needed because categories emerge so fast. Emergent fit implies that existing categories do not have to be ignored and only new categories can be discovered. Existing categories can be “emergent-fitted” to the data.

The researcher found the initial process of categorisation to be quite subjective and difficult. But once the technique was fairly well developed, one of the best ways to validate that the categories do fit the data was to test the categories against a set of theoretical and real case studies. Irregularities were picked up and corrected and the technique tested again until all of the case studies tested gave no problems.

2. Relevance

Relevance implies that the theory should be relevant to some stakeholder. This integrative technique, and its implied theory, is relevant in that it currently addresses a major problem in business and ISD modelling.

3. Work

Work means that a theory “… should be able to explain what happened, predict what will happen and interpret what is happening in an area of substantive or formal inquiry,” (Glaser, 1978:4).

Because the grounded approach was only used to do the data analysis, this criterion is not met and it does not have to be met to comply with the requirements of design science research.

4. Modifiability

Modifiability became important to Glaser (1978) and his students over the years as they have generated many grounded theories. As new things emerge, there is a need for qualifying what came before, while is also a desire to hold on to the existing theory. Glaser’s (1978:5) conclusion is: “… that generation is an ever modifying

process and nothing is sacred if the analyst is dedicated to giving priority attention to the data." Modifiability then measures how quickly the theory can be modified to help explain surprising or new variations.

This criterion of modifiability is difficult to evaluate unless "surprising or new variations" are found and the theory is then modified.

9.3.2 Design science research evaluation

To evaluate the design science research areas of this study, the eight components of an IS design theory by Gregor and Jones (2007) are used (see Table 9-1). They argue for design knowledge to be expressed as theory, because it will ensure that IS rise above the level of a craft and it will provide for a "sounder basis for arguing for the rigour and legitimacy of IS as an applied discipline" (Gregor and Jones, 2007:314).

Component	Description
Core components	
1) Purpose and scope (the *causa finalis*)	"What the system is for." The set of meta-requirements or goals that specifies the type of artefact to which the theory applies, and that also defines the scope, or boundaries, of the theory.
2) Constructs (the *causa materialis*)	Representations of the entities of interest in the theory.
3) Principle of form and function (the *causa formalis*)	The abstract "blueprint" or architecture that describes an IS artefact, either product or method/intervention.
4) Artefact mutability	The changes in state of the artefact anticipated in the theory, i.e. what degree of artefact change is encompassed by the theory.
5) Testable propositions	Truth statements about the design theory.
6) Justificatory knowledge	The underlying knowledge or theory from the natural, social or design sciences that gives a basis and explanation for the design (kernel theories).
Additional components	
7) Principles of implementation (the *causa efficiens*)	A description of processes for implementing the theory (either product or method) in specific contexts.
8) Expository instantiation	A physical implementation of the artefact that can assist in representing the theory both as an expository device and for purposes of testing.

(Gregor and Jones, 2007:322)

Table 9-1: Eight components of an IS design theory

The eight components are discussed in more detail and illustrated by one of the examples used in the article by Codd (1982)) on relational database models for database design (Gregor and Jones, 2007). Then the theory that emerged from the study is evaluated by using the eight components.

1. Purpose and scope

This aspect defines the meta-requirements not for one instance only, but for a class of artefacts. For example, Codd's relational model is applicable to the design of large databases (not single-file structures) accessed by many people.

For this research, the purpose and scope is related to business modelling that can be used as the input or source for further ISD modelling.

2. Constructs

The representation and clear definition of constructs (entities or units of interest in the theory) are considered to be the most basic level in any theory. For instance, in Codd's relational model an n-ary relation is used to represent a relational database table.

In this study, the base, structure and role modelling entities form the basic constructs of the theory and are clearly defined in section 7.3.

3. Principles of form and function

Fundamentally, "form" refers to the artefact's constructs and their relationships (structural properties), while "function" refers to how these are used to achieve the purpose of the artefact (functional properties). For instance, Codd describes both how relational tables are structured and related as well as how they can be used to access and manipulate data.

In this research, it is shown how the base modelling entities are related in the structure modelling entities to form the building blocks of this technique. These modelling

building blocks can be used to communicate business-related analysis and design information.

4. Artefact mutability

IS artefacts are seen to have a special feature in comparison with other types of artefacts, namely their almost constant state of change. Therefore, it is seen as essential to specify the degree of mutability of designed artefacts, including the expected level of adaptation or evolution. For example, in relational database design a main objective is to minimise the effect of changes to the internal representation of data on users and application programmers.

In this study, the form (structural aspects) is considered to be fairly immutable, while the function is considered to be very mutable. New base entities and structure entities are unlikely to be developed much more. Role entities can see some growth when used and could increase and improve to some extent. Applying the technique to specific situations is the area where almost unlimited change can occur. An entire area of research can be done purely on considering the rules that can be generated for specific situations. For instance, if a model phrase refers to the movement of physical objects, a set of specific questions can be asked to enrich the analysis and design, such as “from where?”, “to where?”, “by means of which transporting device?” and “by which deadline?”

5. Testable propositions

A design science theory can give rise to two kinds of testable propositions or hypotheses: more general, more quantitative, algorithmic propositions that can be tested by means of observation, and statistically and less general heuristic propositions that can be tested by design example for a specific problem. Generality is a particular IS research problem not only related to design science theory. For instance, in relational modelling it is stated that relational databases can perform as well as non-relational databases.

In this study, a certain set of specific base entities and their relationships are stated. These can be tested and if other entities and other relationships can be identified, it will show that the original theory was incorrect and can be improved.

6. Justificatory knowledge

These are the "micro-" or kernel theories that inform design products and processes. These theories can be derived from natural science, social science and other design theories and practitioner-in-use theories. According to Gregor and Jones (2007), these theories are the linking element for many, if not all, other aspects of design theory. For instance, relational modelling theory was based on set theory and, interestingly enough, also on human cognitive processes, because it aimed to ease the complex reasoning processes by programmers needed to handle repeating groups of data.

In this study, a grounded theory approach to qualitative data analysis and the theory of linguistics were used as kernel theories.

7. Principles of implementation

This involves how the design is brought into existence by agents performing actions that link processes and products. For instance, in relational modelling, guidelines are provided on how to create a relational database through normalisation.

In this research, some preliminary guidelines are provided on how to use this technique in a number of typical business modelling situations.

8. Expository instantiation

This involves realistically implementing or illustrating the possible implementation of a viable artefact. A major question is whether the physical instantiation is part of the abstract theory or separate from it. Gregor and Jones (2007) argue for considering the instantiation of a part of the theory. For instance, Codd used mock-ups of real systems to help explain working relational databases.

In this study, a possible implementation of this technique using software to assist in the process is illustrated in section 8.3.3.

9.4 Contribution of the research

Introna (1996) proposed a set of principles that can be used to evaluate the contribution of research. These principles are used one by one below to evaluate the contribution of this study.

1. **Does the research raise problems previously not perceived, e.g. problems of an increasing depth, and does it display an ever-increasing fertility in suggesting new problems?**

This research shows that without a clear understanding of ISD modelling entities, there cannot be an integrated approach to modelling, starting from the business side and going all the way to the technical side. Currently no techniques exist to really address modelling from business to technical aspects using only one modelling language.

2. **Does the research anticipate novel facts and auxiliary theories?**

The results of this study indicate that linguistics is a rich but underdeveloped reference discipline for ISD modelling. Most ISD techniques mention linguistics, but do not really take it to its logical conclusion.

3. **Is the research more precise in the assertions and in the facts it explains than previous theories?**

The results of this research can relate all the fundamental ISD modelling entities in use today in various ISD techniques to each other. In this way, it provides a unifying explanation of the relationships between these entities, which does not really exist currently.

4. Has the research unified or connected various hitherto unrelated problems or concepts?

The research done for this study can be seen as a step in integrating business, process, technical and enterprise modelling. Currently all of these areas of modelling have their own sets of modelling techniques.

9.5 Future research

This study has opened up a number of possibilities for future research. These can be divided into the following areas:

(a) Linguistics

A whole area of research is possible around the concepts of language and linguistics. The following are some of the possible research topics:

- Linguistics can be used as an ISD analysis tool. In other words, the documents and interview materials gathered during requirements elicitation can be seen as the "text" to be analysed linguistically. Verbs will indicate actions, personal nouns will indicate agents, nouns will indicate objects/entities and so on. This linguistic analysis can be used to model an information system. The researcher has done something in this vein already in a paper on linguistic analysis and representation of business rules (Joubert, 2009).
- An exciting possibility is having software that can generate analyses and/or models from text input. For instance, providing a policy document for such software can generate a data model and repository, taking the drudgery out of the process for analysts and only employing their higher cognitive skills to assist in the process. Developing an ontology or framework that will link business terms to modelling entities can facilitate the development of such software.

(b) Modelling

This project started out as an attempt to improve modelling. However, it was realised that modelling can only be improved if one knows what to model, hence the current research. Moving from this study, the following are possibilities for research:

- Developing metrics for measuring IS from their models. Only once one has the fundamental modelling entities in place, can one start to answer related questions. Theoretically, the size of an information system can be estimated from its model (obviously, with more and more accuracy as more and more detail is modelled).
- Research into the problematic area of modelling organisations in enterprise architectures.
- Developing conversion software between existing models.

(c) Ontology

This part will most probably contribute the most to the continued study of this research. Athenikos and Song has developed a framework of ontology-based modelling patterns (2008). They fundamentally addressed the same research problem as this study, but followed an ontology-based approach to it.

Traditionally, ontology is the philosophical study of the nature of being and existence. (Kayed and Colomb, 2005). More recently in IS, the term “ontology” has developed to refer to something more specific. Although there is a lot of debate on its definition, one of the most cited definitions is the one by Gruber (1995:908): “An ontology is an explicit specification of a conceptualisation”, where a conceptualisation is “an abstract, simplified view of the world that we wish to represent for some purpose”.

In their most basic form, these ontologies provide shared vocabulary representing a specific domain’s knowledge. They can also be seen as descriptions of the conceptual knowledge of an application domain. Examples of ontologies are the EngMath Ontology for mathematical formulae and symbols, the Tove Ontology for generic enterprise modelling knowledge and the Plintius Ontology of the chemical

composition of ceramic material (Mihoubi et al., 1998). Although ontologies can take many forms, they will always include a vocabulary of terms and some specification of their meanings (Kayed and Colomb, 2005).

Recently, ontology has been used in various information and information technology fields to, for example, solve semantic problems, help in matchmaking, do information discovery, do information retrieval, build explicit reusable knowledge and evaluate existing information system modelling techniques (Kayed and Colomb, 2005).
The logical next step for this research is to formalise the findings into an ontology of business and ISD modelling fundamental entities.

9.6 Concluding remarks

The problem statement of this study is:

The current modelling techniques do not bridge the gap between business and ISD.

The main research question is:

Can an integrative modelling technique be developed to bridge the gap between business and ISD?

It can be answered in summary as follows:

Most current business modelling techniques suffer from one of two problems: they are either easily understood by business users (mostly textual), but too simplistic to be used in ISD modelling, or they are usable for ISD modelling (using some existing ISD modelling technique like UML), but then they are too complex for the average business user.

To develop an integrative modelling technique between business and ISD modelling, it is important to first understand what the fundamental modelling entities in business

and ISD are. Using a grounded approach, the study found that these entities follow a linguistic structure and can be divided into three main categories:

- *Base entities (corresponding to the morphological level in linguistics), representing the real-world objects that make up organisations and systems as the words of the proposed modelling language. The high-level base entities are things, actors and objects.*
- *Structure entities (corresponding to the syntactical level in linguistics), using the base entities plus specific language constructions to form model sentences and phrases. The structure entities are models, model views, model sentences and model phrases.*
- *Role entities (corresponding to the semantic level in linguistics), helping to define meaning to base and structure entities in various situations. Role entities can be classified as subject, predicate and complement roles.*

An understanding of the properties and attributes of these entities, together with an understanding of the relationship between these entities, forms a basic method to model business situations both easily and expressively. Furthermore, a software instantiation of the technique can be envisaged that will simplify the process of business analysis and design.

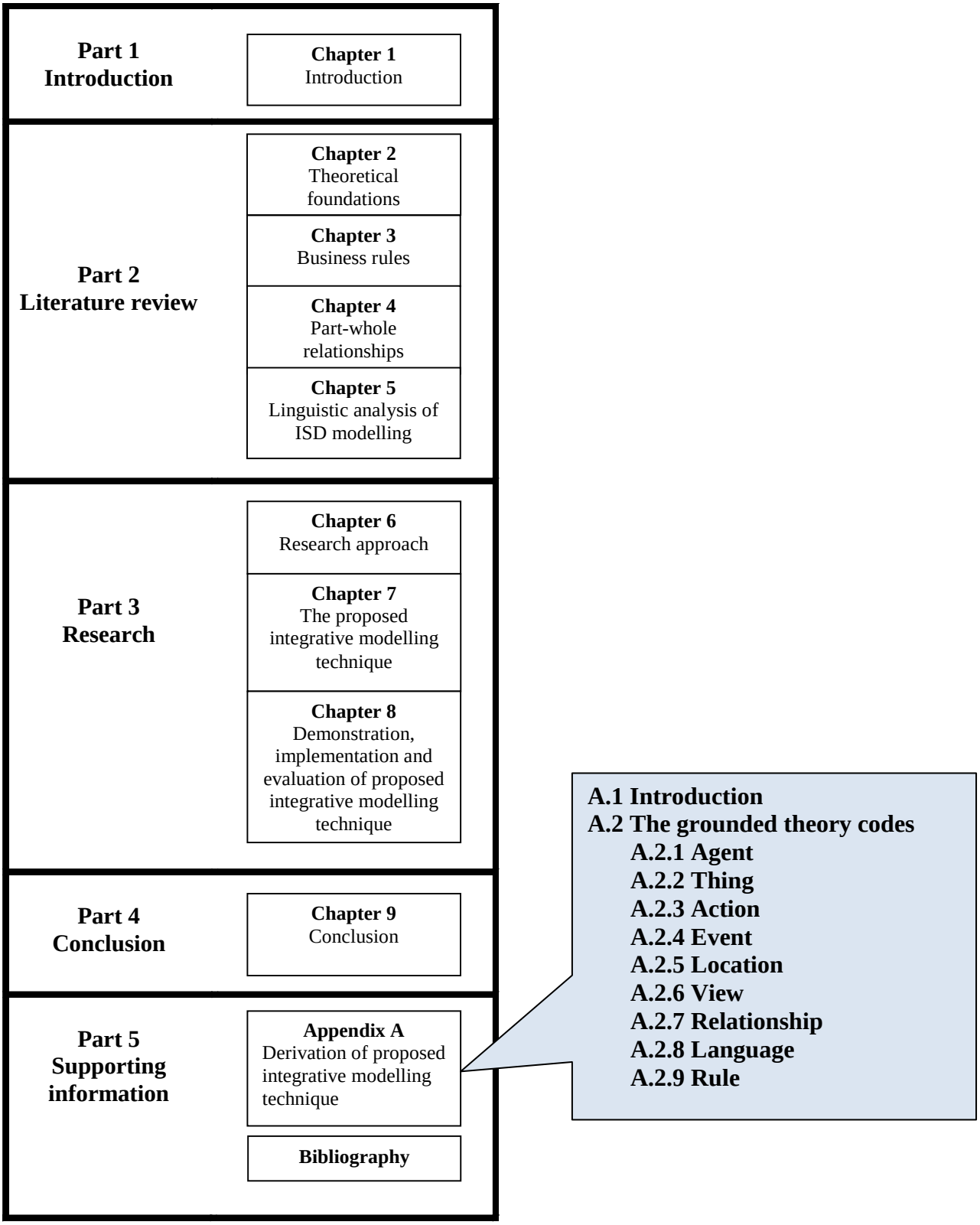
The research done is evaluated from the perspectives of grounded theory and design science research. The contribution of the research is also evaluated.

It is important to note what Glaser and Strauss (1967:32) had to say about theory (specifically grounded theory, but it is also applicable to design science theory): “Our strategy of comparative analysis for generating theory puts a high emphasis on theory as process; that is, theory as an ever-developing entity, not as a perfected product.” This is the reason for choosing the word “*towards*” in the title of this thesis. This study only gives a step in the direction of better understanding business and ISD modelling.

Part 5

Supporting Information

10. Appendix A: Derivation of basic concepts of integrated modelling language



This appendix shows how the link was made between the original source material (the different technique descriptions) and the resultant integrative modelling technique.

10.1 Introduction

Both of the two major approaches to grounded theory (Glaser and Strauss, 1967; Strauss and Corbin, 1998) basically have three high-level steps: coding concepts from the data, categorising and abstracting these codes, and finally developing a theory from these categories of codes.

During the various coding and interpretation steps of the grounded theory process, many codes eventually became a few and the relationships between these codes became increasingly clear. Furthermore, as explained before, the grounded theory process caused the researcher to go back to the literature at various stages to enhance the understanding of some of the concepts that emerged. As a result of this, systems theory, business rules, part-whole relationships and linguistics were studied further as the theoretical foundations of this research. Incorporating these foundational concepts enhanced the findings of research and helped in the very complex process of categorisation. Out of these categorisations a modelling technique eventually emerged.

The result of the code categorisation process is given in the next section, linking the codes back to the original techniques from which they came.

10.2 The grounded analysis codes

The following codes emerged as central codes of this study:

- Agent
- Things
- Actions
- Events
- Locations

- Views
- Relationships
- Linguistics

A detailed description follows of how each of these codes link back to the original techniques and to the theoretical foundations.

10.2.1 Agent

One of the codes that occur in almost all techniques is *agent*. It is mostly called *agent* in the various techniques, but is also called *actor*, *resource*, *organisation*, *participant*, *stakeholder* and *role*. The main aspects of agent derived from the grounded analysis are the *actions* that agents perform; the *relationships* that actors have in such as the *roles* and *responsibilities* they have; and the *types* of agents that can be identified. See Figure 10-1 for a summary of the main codes related to agent.

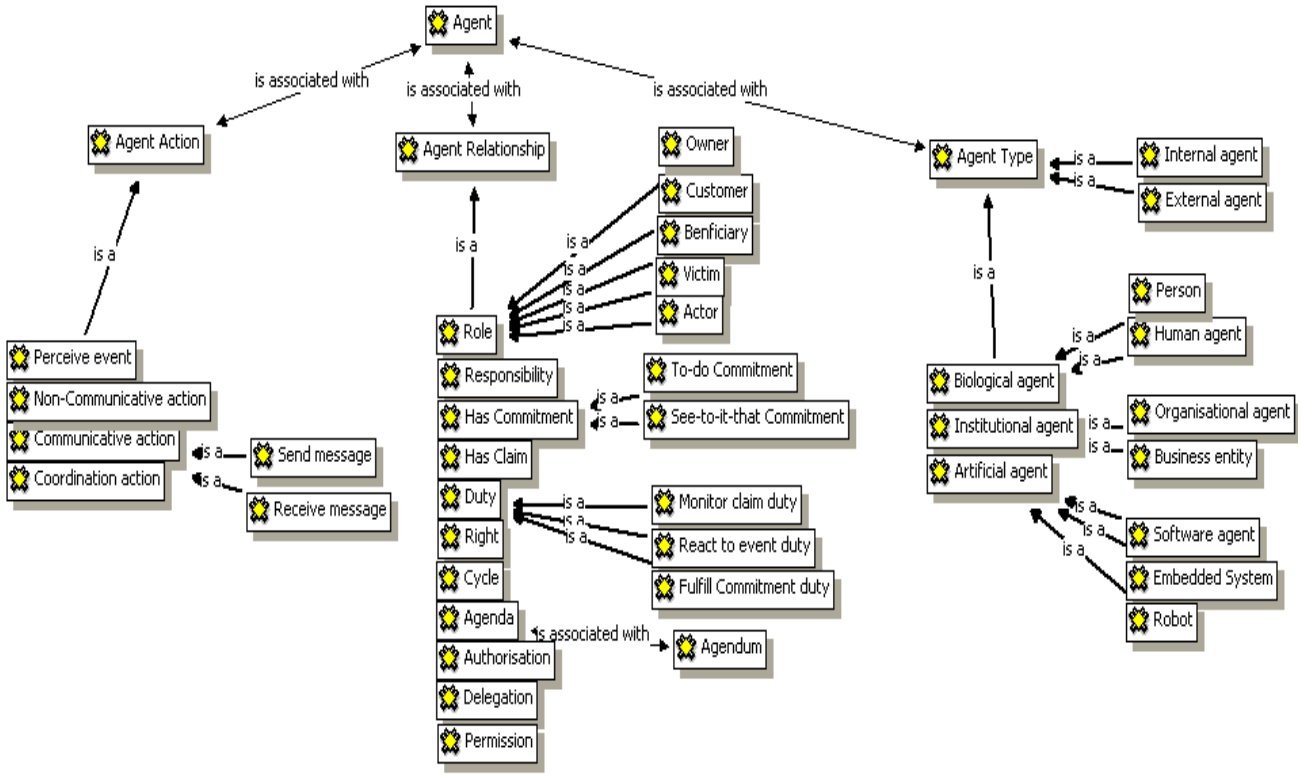


Figure 10-1: Agent-related codes

10.2.1.1 Link to techniques

The way in which *agent* and its related codes are derived from the various modelling techniques is described in the following table:

Technique	Main concepts
ALC/OLC	– There are **agents** and **organisational agents**. – Every agent has a **life cycle,** which has operations and **agent roles** (similar to object states). – Agents have **responsibilities** towards other agents. – Agents discharge their responsibilities by means of **operations**.
AOR	– Three types of agents: **biological/human, institutional and artificial** (software agent, embedded system, robot). – Only agents can perceive events, perform actions, communicate, make commitments and have claims, i.e. **actor actions and relationships**. – **Commitments** are binary relationships between two agents. – **Commitments** arise from certain communicative actions. – Two kinds of commitments: **to-do** (action) commitment, **see-to-it-that** (condition holds) commitment. – When one agent makes a commitment, the other agent has a **claim** on the outcome of the commitment. – An institutional agent consists of other agents, called **internal agents,** which can act on its behalf. It can perceive and act through its internal agents. – An institutional agent can deal with other **external agents**. – Internal agents have **duties** and **rights** (certain permitted actions that an agent can do on behalf of the organisation). – There are three **kinds of duties:** to monitor certain claims, to react to certain events, to fulfil certain commitments. – Internal agents play certain **roles**.
ARM	– **Agency** is a collection of humans that participates in contractual relationships. – **Agencies** employ one or more other persons as agents. – **Complex agents** (macro-agents) are decomposable, lowest (**primitive) agent** must be a person. – Relationships between agents imply **responsibility**.
BPMN	– **Business entities** or business **roles** are types of participants. – Each **participant** has his own viewpoint, but during the execution of the process he can only control his own activities. – A **swim lane** in a **pool** (subpart of pool) represents a **participant** in a process.
DFD	– **External entity** is the source or destination of the information flow. – External entities are only entities that **originate** or **receive** data/resources.

Technique	Main concepts
Gantt and PERT	– A **resource** can either be a type of material or **work**. When it is work, it represents a role or agent.
IDEF0	– **Mechanism**; the means to produce output; can include agents.
IDEF1	– Agent concept not specifically addressed, but an **entity** can be an agent, among other things.
IDEF1X	– Agent concept not specifically addressed, but an **entity** can be an agent. – People can be a **source** of information.
IDEF3	– Agent concept not specifically addressed.
IDEF5	– The concepts of **kind** and **term** also include the concept of agent. – Identifies certain case relations, like **agent**-instrument, **agent**-object, action-**recipient** and **agent**-action (see relationships for more information).
LAP	– An **organisation** is a system of **human beings** with a particular **purpose** or **mission.** – **Actors** are the motors of an organisation. – **Actors** constantly loop through the **actor cycle** in which they deal with their **agenda** (to-do-list). – An **agendum** is a coordination fact to which the actor is committed to respond. – **Actor role** is the "amount" of authority to perform particular acts. – Three types of **role assignments** to subjects: **authorisation** (fairly permanent), **delegation** (transfer of authorisation), **propagation** (transfer of authorisation to embedded transactions). – One **subject** may fulfil a number of **actor roles** concurrently and/or successively. An actor role may be fulfilled by many subjects concurrently and/or successively. – **Organisations** can be seen as social systems consisting of human beings who are **social individuals**. – **Social individuals** or subjects can perform **production, coordination** or **communication acts.** – **Social individuals** perform two kinds of acts: **production acts** bringing about goods and/or services, and **coordination acts** entering into and complying with commitments and agreements to perform production acts, i.e. requesting or promising a production. – The actor who starts a transaction is called an **initiator** or a **customer.** – The other actor is called the **executor**, **producer** or **supplier.** – **An actor's role** may be fulfilled by many subjects **concurrently** as well as **collectively.** – One **subject** may fulfil a number of actor's roles **concurrently** and **successively.**

Technique	Main concepts
ODP	– An **agent** is an object fulfilling one or more **agent roles.** – An **agent role** is a service (set of actions) offered by the agent to its environment (other agents). – Structuring rules include the following: o **Obligation** – prescription that a particular behaviour is required o **Permission** o **Prohibition** o **Policy** – a set of rules related to a particular purpose
RAD	– **Role** is a particular **responsibility** which involves a sequence of **activities.**
SQL	– Certain **roles** have specific **permissions** to perform various **database actions** on **database objects.** – **Permission** is granted, revoked or denied for a certain **role** to access certain database objects or to perform certain database-related actions.
SSM	– The root definition (CATWOE) consists of the following: o **Customer/client** – who benefits (or **beneficiary**/**victim**) o **Actor** – who facilitates transformation o Transformation – the process o *Weltanschauung* – the worldview of the stakeholders o **Owner** – to whom the system is answerable o Environment – A **system** has the following: o Ongoing purpose o A decision-taking process – this makes it an **agent** o Components that are also systems (i.e. subsystems) o Components that interact o An environment o A boundary o Resources o Continuity
UML	– **Actors** (in use case diagram) – a **role** played by a **person, system** or **device** that has a stake in the successful operation of the system. – A **role** is a type; not a particular **person/system/device.** – An **object/class** can also represent an agent. – Use case identifies interaction between **actors** and use case in the form of a dialogue.
Zachman	– **Who** column/focus. The types of human resources that are needed to initiate or perform an activity.

10.2.1.2 Link to theoretical foundations

The grounded concepts of *human agent* have some relationships to the systems concept. The Magee and De Weck (2004) taxonomy (section 2.6) gives some insight into human agents. The human control attribute specifies systems as being autonomous, having a human in the loop, or being mixed. The human wants attribute specifies shelter, food, transportation, communication, security, longevity and health, entertainment, aesthetic pleasure, education, and social, emotional, spiritual and curiosity aspects.

The grounded concepts of *artificial agent* and *institutional agent* both exhibit all of the characteristics of systems. For example, a cellphone (plus its supporting system[6]), an *artificial agent*, has an ongoing purpose of providing mobile communication at a fee, while the *organisational agent*, restaurant has the ongoing purpose of providing food at a price.

Furthermore, an *artificial agent* can be classified as a human activity system with mostly designed system components. It can also be classified as an abstract system with concrete and conceptual components. An *institutional agent*, on the other hand, can be classified as a human activity system with designed as well as social and cultural components. It can also be classified as an abstract system with concrete and conceptual components.

Furthermore, *artificial agents* can be classified as somewhere between open and relatively closed systems in how they interact with their environment; mostly between autarchic and symbiont systems (artificial intelligent systems will rank higher) in how they cope with environmental changes; and mostly between automatic-sequential and controlled-regulated in how they respond to their environment. On the other hand, the grounded concept's *institutional agents* can be classified as open systems in how they interact with their environment; somewhere between symbiont and heuristic systems

[6] An artificial agent like a cellphone can never be seen as only consisting of its hardware and software. The supporting system (humans, organisations and procedures) around the hardware and software is an essential component of any artificial agent. A cellphone without the supporting network to phone other people will cease to be a cellphone!

in how they cope with environmental changes; and mostly between adaptive-self-optimising and self-learning-self-organising in how they respond to their environment.

An *artificial agent* can be classified as mostly between an externally governed system and a system with embedded goals and controls with respect to self-government; mostly between rigidly controlled and purposive systems when considering self-organising behaviour; and somewhere between simple dynamic and multilevel when considering intrasystem and system-milieu interactions. An *organisational agent* can be classified as mostly between a self-learning system and a system with multiple deciders with respect to self-government; mostly between purposive and purposeful systems when considering self-organising behaviour; and somewhere between multilevel and evolutionary when considering intra-system and system-milieu interactions.

10.2.2 Thing

Another one of the codes that occurs in almost all techniques is *thing*. It is mostly called *entity* or *object* in the various techniques. The reason for giving it a totally different name is because both the terms “entity” and “object” have very specific meanings in ISD modelling. These terms are mostly related to data and object modelling respectively. The main aspects of *thing* derived from the grounded analysis are the *properties* of things; and the *types* of things that can be identified. See Figure 10-2 for a summary of the main codes related to *thing*.

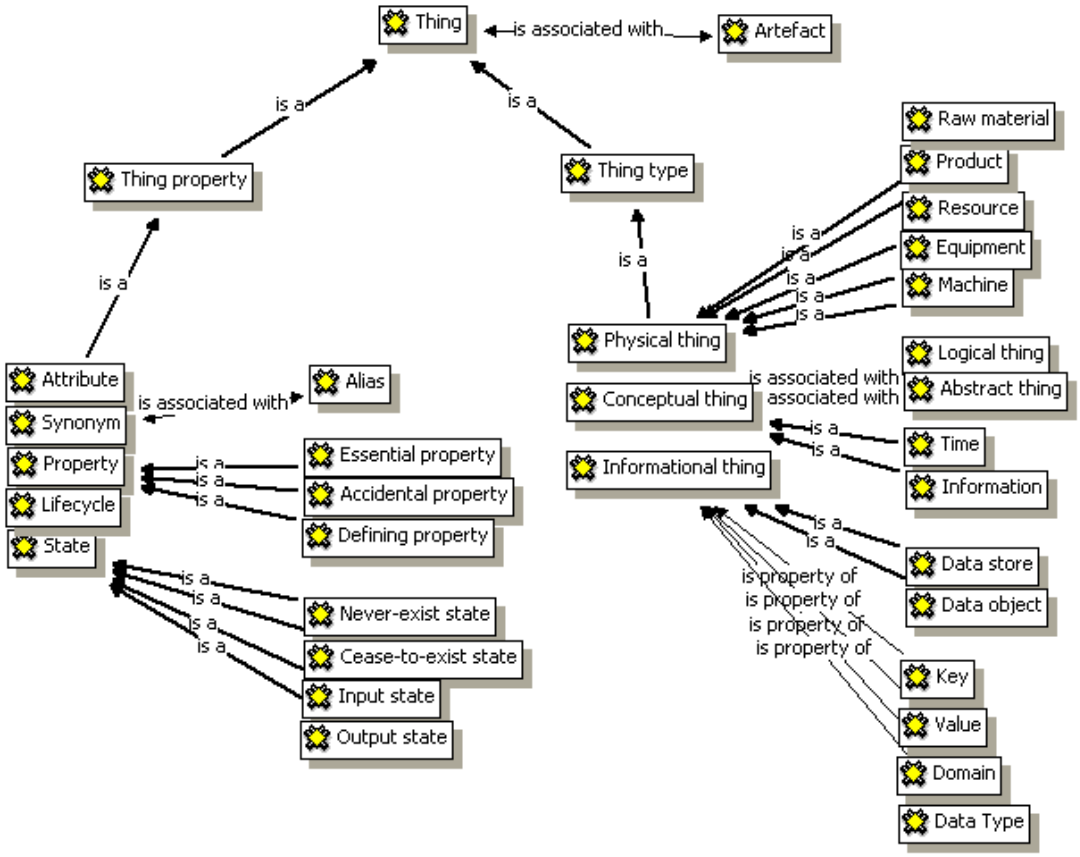


Figure 10-2: Thing-related codes

10.2.2.1 Link to techniques

The way in which *thing* and its related codes are derived from the various modelling techniques is described in the following table:

Technique	Main concepts
ALC/OLC	– **Objects** have **states.** – Objects have **object life cycles** (OLC). – Two special states are **never-exist** (start) and **cease-to-exist** (end). – Every complete **OLC** start from the never-exist state and end in the cease-to-exist state. – Operations either **change the state** of an object or **keep the state** of an object steady.
AOR	– An application domain consists, among other things, of ordinary **objects.** – Information systems have to represent **entities** that occur in the universe of discourse. – Entities have **properties.**

Technique	Main concepts
	– Entities are classified by means of **entity types.** – Each entity type defines a list of (**stored** and **derived**) attributes. – Together, the value of all **attributes** of an entity form the **state** of it.
ARM	– **Objects** are of two types: **physical** (tangible entities) and **logical** (e.g. time, information). – Objects are, among other things, **resources** that agents utilise.
BPMN	– A **data object** provides information to activities. – The information in a data object can be **text** or **graphical.**
DFD	– A **data store** is a holding place for information. – **Information** can either be written, verbal or electronic. – A data store can be **permanent**, **manual** or **transient** (deleted after processing). – When resource flow is considered, **material resources** are involved.
Gantt and PERT	– None.
IDEF0	– Data and objects are represented by **nouns** or **noun phrases.** – Systems consist of, among other things, **information, software, equipment, products, raw materials, machines.** – **Systems** are automated and non-automated.
IDEF1	– Information model has two parts: a **structural** part (diagrams) and **dictionary.** – An entity has **properties/characteristics.** – An entity can be real, **physical** or **abstract.** – An **entity class** is a class of individual member entities. – An **attribute** is an individual **property** of an entity and has both a **name** and a **value.** – A **key** is a unique combination of attributes and values. – Keys can be **single** or **compound.** – There are **alternate** keys.
IDEF1X	– Basic constructs are **things** about which **data** is kept, **relationships** between these things, **characteristics** of these things (**attributes**). – An entity can have **synonyms/aliases/non-standard names.** – An entity represents things of interest **real** or **abstract.** – Entities have keys: **primary key, foreign key, candidate key, alternate key.** – For an entity instance, a specific attribute must have a value or **null value.** – Attribute values have a **domain.** These domain classes can be immutable vs. time-varying; finite/fixed (e.g. USA states) vs. possibly infinite (e.g. surnames). – Can distinguish a **base** and **type** domain. **Base domain** has basic **data types** like integer, text, binary, etc. The domain rule is to only allow acceptable types. **Type domains** are subtypes of base domains with further constraints, form a

Technique	Main concepts
	hierarchy of domains, not necessarily mutually exclusive, complying with parent rules as well.
IDEF3	– An **object** is any **physical** or **conceptual thing** recognised and referred to by participants in the domain. – Objects are often **nouns/noun phrases** that can be coupled with a **state** descriptor (e.g. Purchase Order: Approved). – A state can have condition types: o **State** – those conditions necessary for an object to be in a state (e.g. water: frozen → temperature <= 0^{o}C). o **Exit** – those conditions sufficient for an object in a given state to cease being in that state. o **Transition** – apply to the "interface" between a state and an outgoing link. o **Entry** – conditions sufficient for an object to enter the state, given an object in the source state. – Makes a distinction between **facts** (what is) and **constraints** (what should be). – **Constraints** can be conditional or absolute.
IDEF5	– **Terms** denote **objects**. – Kinds are not **types** or **classes.** Although all three are "categorical", they indicate some grouping of **individuals** into categories. – All three are **instantiable** – different **individuals** can be instances or members of the same group. – Types and kinds are **intentional** – the identity of a type or kind is not dependent upon its membership (i.e. it can grow over time). – **Kind**, traditional definition: for every kind K there is a set N of **properties** that are individually necessary and jointly sufficient for being a K, i.e. x is a K if and only if x has every property of K. – Compare **essential** vs. **accidental properties.** – **Kind** is an objective category of objects that are bound together by a common nature. – Enterprise ontologies have a more flexible notion of kind. They use the term "**defining properties**" instead of nature. – Some defining properties can be accidental. Defining and essential properties are orthogonal. – For every x of a kind K, x has at least one of the defining properties of K. – An **attribute** is a mapping that takes each member of a given set of individuals to a single specific value, e.g. colour-of. – A **property** is a characteristic of things, e.g. being red. – **Characteristic** is a neutral term encompassing both attribute and property. – An **ontology** is a catalogue of terms used in a domain. – Related terms are **vocabulary**, **taxonomy** and **terminology.** – A **term** is a definite descriptor that refers to an object or

Technique	Main concepts
	situation-like thing in the real world.
LAP	– The result of performing a production act is a **production fact.** – The result of performing a coordination act is a **coordination fact.** – Both **material** and **immaterial facts** come into existence when corresponding coordination acts are performed.
ODP	– An artefact is an **object** fulfilling an artefact role. – An **artefact role** is a role involving the use of **resources**, but not able to initiate actions with respect to those resources. – An **artefact role** is a service offered by an artefact.
RAD	– A token indicates a change of **state.**
SQL	– The following are database objects: o **Schema** contains a database definition (metadata). o **Identity** is an autonumber. o **Table** stores data, can be temporary. o **View** is a virtual table. o **Constraints** (applicable to tables or columns): ▪ Foreign key and references ▪ Check (domain integrity), e.g. min_level <= 10 ▪ Unique ▪ Primary key ▪ Null, Not Null – A table has **rows** and **columns**. – An **entity** is the **logical** and a table is the **physical** representation of data. – A column represents one **attribute** of an entity/table. It has a specific **data type.** – The following are some of the possible **data types:** o **Exact numerics** like bigint, int, bit and money o **Approximate numerics** like float and real o **Date time** o **Character string**
SSM	Nothing specifically related to thing.
UML	– A **class** vs. an **object** is like template/mould vs. instance; or dictionary word vs. actual thing. – **Class** defines what can be, **object** what is. Class defines the rules, object expresses the facts. – An object has a **life**, events that trigger changes in the object's state. – An **object** knows the following: o Its current **state/condition** – properties at a specific time. o What it can do. o What can be done to it. o About itself – its **properties.** – Encapsulation means exposing an object's **interface** and **purpose,** but hiding **data** within (structure and state) and **implementation.**

Technique	Main concepts
	– A **package** is a place to put things, the UML version of a directory. – **Attributes** of a class describe its appearance and knowledge. An attribute has: o Name o Data type: primitive, language supplied or abstract, developer defined o Rules constraining input values o Default value o Visibility: public, private, protected, package – The **state** of an object is its current condition and is normally described with an **adjective** like open or closed account. – A **software component** can be an executable, file, document, table or software library. – Each **node** is a **physical object** that represents a processing resource. – Two types of elements: nodes (**resources**) and **associations** (connections).
Zachman	– **What/data** – physical and conceptual things that are important to business.

10.2.2.2 Link to theoretical foundations

The Magee and De Weck (2004) taxonomy (section 2.6) gives some insight into things. The degree of complexity attribute compares the different levels into which physical things can be divided. The degree of complexity goes from part to sub-assembly to machine to equipment. The functional type attribute describes various operands and the Matter (M) operand, consisting of physical objects and organisms, defines physical things, while the Information (I) and Value (V) operands define informational things.

10.2.3 Action

One of the codes that occur in almost all techniques is *action*. It is called many names in the various techniques, such as *process, task, activity, act, service, transformation* or *transaction*. The main aspects of *action* derived from the grounded analysis are *categories* of actions, the *levels* of action, the *purposes* of actions, the *properties* of actions and the *types* of actions that can be identified. Please note that a major aspect

of actions, namely *action relationships*, will be discussed later under the heading of relationships. See Figure 10-3 for a summary of the codes related to action.

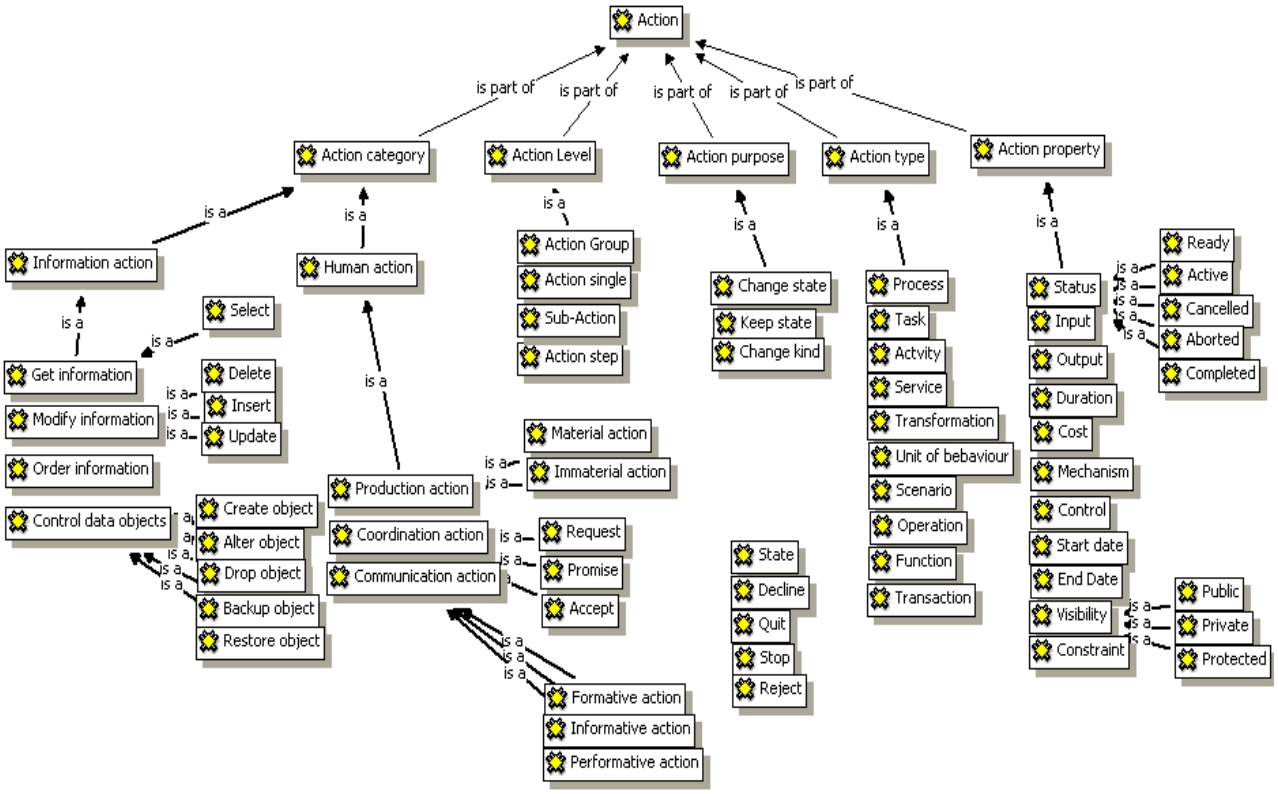


Figure 10-3: Action-related codes

10.2.3.1 Link to techniques

The way in which *action* and its related codes are derived from the various modelling techniques is described in the following table:

Technique	Main concepts
ALC/OLC	– **Operations** are either **changing** or **maintaining** the **state** of affairs. – An object is transformed from state to state by an **operation.** – **Agent life cycle operations** are actions through which agents discharge their responsibilities to other agents.
AOR	– There are **social interaction processes** involving agents and **non-social interaction processes** involving agents and their inanimate environments. – There are six **relationships** in which agents, but not objects, participate: o Perceive environment events o Receive and send messages

Technique	Main concepts
	○ Do non-communicative actions ○ hasCommitment ○ hasClaim – **Commitment** is a specific action to be performed in due time. Two types: **to-do** and **see-to-it-that.** – Commitment **operations** are: ○ Creation of commitment ○ Waiving a claim (releasing from a commitment) ○ Assigning a claim to another agent – **Claim** is a specific event that ought to happen in due time. – Commitment **operations** are: ○ Creation of a commitment ○ Cancellation of a commitment ○ Waiving a claim (releasing from a commitment) ○ Delegation of a commitment ○ Fulfilling a commitment
ARM	None for action
BPMN	– A **business process** is an activity performed within or across companies and organisations. – Three types of **business processes:** ○ **Private**, internal (workflow) ○ **Abstract**, public ○ **Collaboration**, global – Every process has a **name, type, status** and other **attributes.** – The **status**es for a process can be *ready*, *active*, *cancelled*, *aborting*, *aborted*, *completing* or *completed*. – Each participant has own viewpoint, but during **execution of process** can only control own activities. – One type of flow object is an **activity,** the work done by a company. – Activities can either be **processes**, **subprocesses** or **tasks.**
DFD	– A **process** must have **input** and **output.** – The **inputs** and **outputs** must be balanced, i.e. the inputs must have enough information to create the outputs. – A process represents the **flow** of data or material resources through a system.
Gantt and PERT	– A **project** is broken down into **activities.** – Every activity has as a minimum a **start date**, **end date** and **duration.** – Activities can also show **resources, costs,** and **percentage complete.** – **Resources** are associated with a project activity. – **Resources** are utilised in the execution of activities.
IDEF0	– Two primary modelling components are **functions** and the data and objects that interrelate functions. – Functions are equivalent to **activities** and **processes.** – **Input** is **data** or **objects** that are transformed or consumed.

Technique	Main concepts
	– **Output** is data or objects produced by the process. – **Controls** are the conditions required to produce output.
IDEF1	– None
IDEF1X	– Relationships are expressed as **verbs.**
IDEF3	– Objects participate in **processes.** – Real-world processes consist of **units of behaviour:** happenings, events, decisions, acts, processes. – **Scenario/story** describes a recurring situation (or set of situations) that describes a typical class of problems addressed by an organisation or system. – A **scenario** is a sequence of activities within the context of a given situation.
IDEF5	– **Processes** involve two sorts of change: **change in state** (water from frozen to solid) and **change in kind** (incineration from wood to ashes). – **Processes** can be thought of as special kinds, but they also occur over an interval of time.
LAP	– **Production acts** are either material acts (manufacturing, storage and transportation) or immaterial acts like court judgment, granting a decision and appointing someone. – **Coordination acts** are brought about by a sequence of **communication acts: formative** acts (express, transmit, receive), **informative** acts (inform, confirm) or **performative** acts (agree, commit). – An actor can be addressed by another actor as the addressee of a **coordination act** that the other actor wants to perform. – A specific type of interaction is a **transaction,** which is a finite sequence of coordination acts between two actors concerning the same production act.
ODP	– A service is a **set of actions** taken by the agent, possibly involving one or more **resources.** – To find services, **yellow** and **white page** functions are provided. – The following types of **functions** should be provided to enable distributed processing: security, repository, coordination and management.
RAD	– **Activities** that can be decomposed further.
SQL	– Certain **actions** can be done on data objects (e.g. database, table, index, trigger): o **Create** new object. o **Alter** an existing object, e.g. add columns to a table. o **Drop** an existing object. o Make a **backup** of object. – Certain **actions** can be done on data inside a table or view: o **Select** or read data records/rows. o **Insert** or add new records to a table. o **Delete** data records.

Technique	Main concepts
	○ **Update** data records. – These **actions** done on data inside a table or view can be defined in more detail by specifying: ○ Which tables or views (“from”, “join”) ○ Which records (“where”, “union”) ○ In which order (“order by”) ○ Which groups (“group by”, “having”) ○ **Functions** like “data and time”, “mathematical”, “string” and “image” – A **transaction** is a single unit of work that must be executed as one and then **committed**, or if any subpart fails, the whole transaction must be **rolled back.**
SSM	– Part of CATWOE is a **transformation** from **input** to **output.** – CATWOE can also be expressed as **PQR:** “Do P by Q in order to contribute to achieving R.” – It answers three questions: what to do (P), how to do it (Q) and why it is being done (R).
UML	– Use case describes **steps** in a **dialogue,** a step-by-step description of the **conversation/interaction** between the system and user. – Activity diagram describes **activities** that are linked together by means of **conditional logic.** – Activity diagrams can represent **sequential** as well as **concurrent** activities. – **Activity** diagrams have the following parts: activities, guard conditions, decisions (mutually exclusive), merge points, start and end points, concurrency, fork, synchronisation and transitions. – In a class diagram, an **operation** defines the **behaviour** of a class. – An **operation** has: ○ Name ○ Input parameters ○ (Optional) return data type ○ Visibility (public, private, protected, package) ○ (Optional) class level operation ○ (Optional) constraints, rules that must be enforced in the execution of the operation
Zachman	– **How**/function, all actions performed by the business.

10.2.3.2 Link to theoretical foundations

According to Magee and De Weck (2004), five basic actions can be executed on the operands Matter (M), Energy (E), Information (I) and Value (V). The five basic actions are as follows:

- **Transform** or **process** - transform objects into new objects
- **Transport** or **distribute** – change the location of objects
- **Store** or **house** – provide buffers in the network by holding objects over time
- **Exchange** or **trade** – exchange objects mainly via the Value operand
- **Control** or **regulate** – drive objects from some actual state to a desired state

10.2.4 Event

One of the codes that occur only in a few techniques is *event*. It can possibly be seen as an attribute of action, but more careful consideration shows that it is an important separate modelling entity. It is called mostly *event* in the various techniques and is sometimes also called *trigger*. The main aspects of event derived from the grounded analysis are the *properties* of events, and the *types* of events that can be identified. See Figure 10-4 for a summary of the codes related to event.

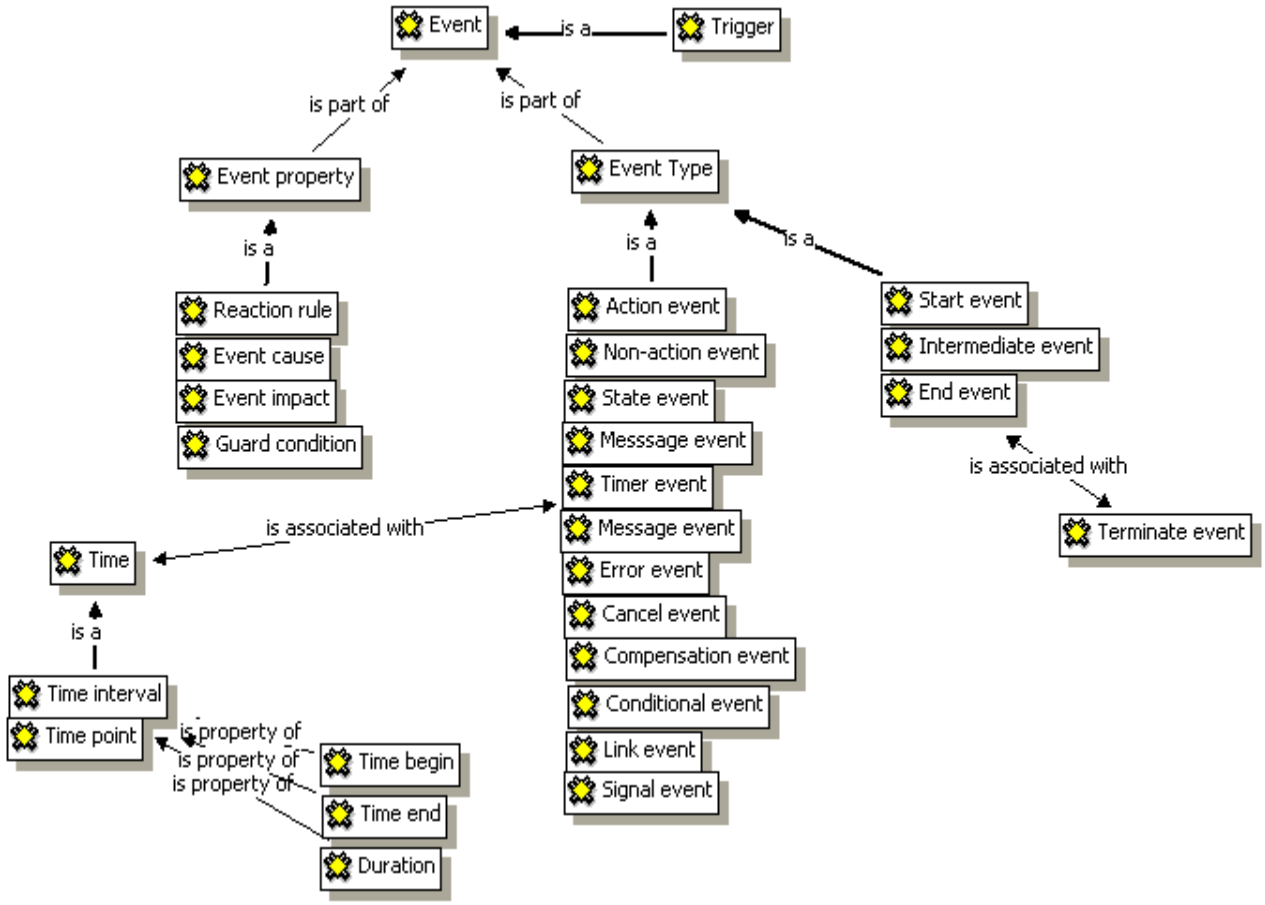


Figure 10-4: Event-related codes

10.2.4.1 Link to techniques

The way in which *event* and its related codes are derived from the various modelling techniques is described in the following table:

Technique	Main concepts
ALC/OLC	– None
AOR	– **Claim** is a specific event that ought to happen in due time.
ARM	– None
BPMN	– An **event** is something that happens during a business process and affects the execution of it. – Events affect **process flow.** – Events usually have **cause** (trigger) or **impact** (result). – Events can be **thrown** (create a result) or **caught** (react to a trigger). – Three groups of **events:** Start, intermediate and end. – The types of events are:

Technique	Main concepts
	o **Message** from a specific participant (agent or other action) o **Timer,** a specific time-date or a specific cycle o **Error** events thrown in children actions o **Cancel** event in a transaction o Activating and performing **compensation** o An event that is triggered when a **condition** become true o **Link** two connecting sections of an action o A **signal** arrives that has been broadcast from another action o **Terminate** ends actions immediately without compensation or event handling o **Multiple** triggers can be assigned to an action
DFD	– None
Gantt and PERT	– None
IDEF0	– None
IDEF1	– None
IDEF1X	– None
IDEF3	– None
IDEF5	– None
LAP	– The creation of a fact of any type is a **state transition** in one of the two worlds (physical or social). – An **event** is a particular transition at a particular time.
ODP	– None
RAD	– A **trigger** is the initiation of a new role.
SQL	– A **trigger** is a stored procedure (a group of database actions with a specific name and parameters) that executes when certain specific actions take place.
SSM	– None
UML	– A use case identifies its initiation or **trigger,** either actor action, time or system event (e.g. error condition, device signal). – One of the elements of a state chart diagram is an **event** (internal or external). The following types of events are identified: o **Time event** – evaluates the passage of time as a trigger. o **Guard condition** – controls the response to an event; when event occurs, the condition is tested. o **Call event** – most common invocation of an operation on the receiving object. o **Change event** – tests for a change in the object or a point in time. o **Send event** – one object tells another object what to do.
Zachman	– None

10.2.4.2 Link to theoretical foundations

No specific links.

10.2.5 Location

One of the codes that occur only in a few techniques is *location*. The code *node* is used mostly in the various techniques. The reason for not using *node* as the name is because *node* has a very specific meaning in ISD modelling, namely a location in a network. The term *location* is a much wider term, including the concept of a node. The main aspects of event derived from the grounded analysis are the *properties* of locations, and the *types* of location that can be identified. See Figure 10-5 for a summary of the codes related to *location*.

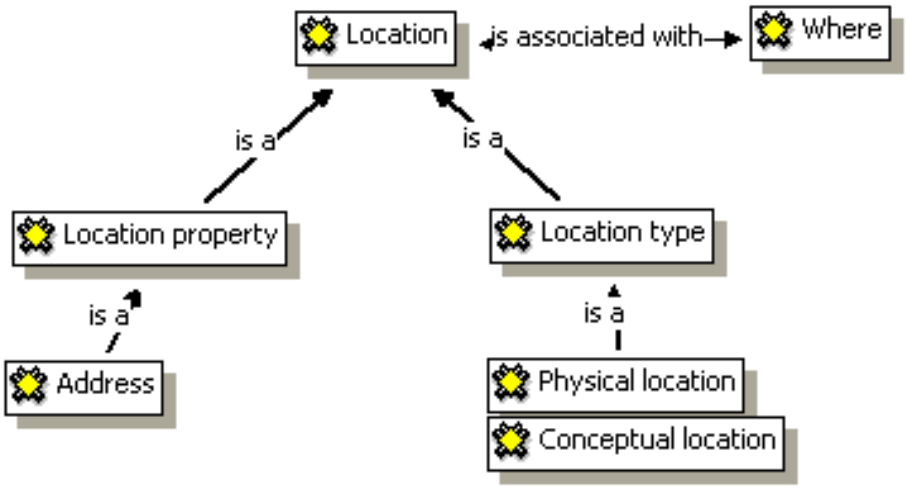


Figure 10-5: Location-related codes

10.2.5.1 Link to techniques

The way in which *location* and its related codes are derived from the various modelling techniques is described in the following table:

Technique	Code
ALC/OLC	– None for location
AOR	– None
ARM	– None
BPMN	– None
DFD	– None
PERT and GANTT	– None
IDEF0	– None
IDEF1	– None
IDEF1X	– None
IDEF3	– None
IDEF5	– Node
LAP	– Node
ODP	– A **node** is a single unit at a location in space, e.g. a single PC. – A node contains the following: o A set of **capsules.** A capsule is a configuration of objects forming the smallest unit of independent failure. o A **nucleus** that provides a node management function, e.g. an operating system.
RAD	– None
SQL	– None
SSM	– None
UML	– In a deployment diagram, a **node** is a **physical object** that represents a processing resource. – A **node** is mostly a computer, but can be a human resource for manual processing. – Each **node** contains or is responsible for one or more **software components** or objects. – Two types of elements: nodes (**resources**) and **associations** (connections). – Deployment diagrams can also function as **network** diagrams.
Zachman	– **Where**/network, all locations or places where activities are performed or things stored.

10.2.5.2 Link to theoretical foundations

No specific links.

10.2.6 View

One of the codes that occur in almost all techniques is *view*. It is called mostly *view* or *perspective* in the various techniques. The main aspects of event derived from the

grounded analysis are the *types* of views that can be identified. See Figure 10-6 for a summary of the codes related to view.

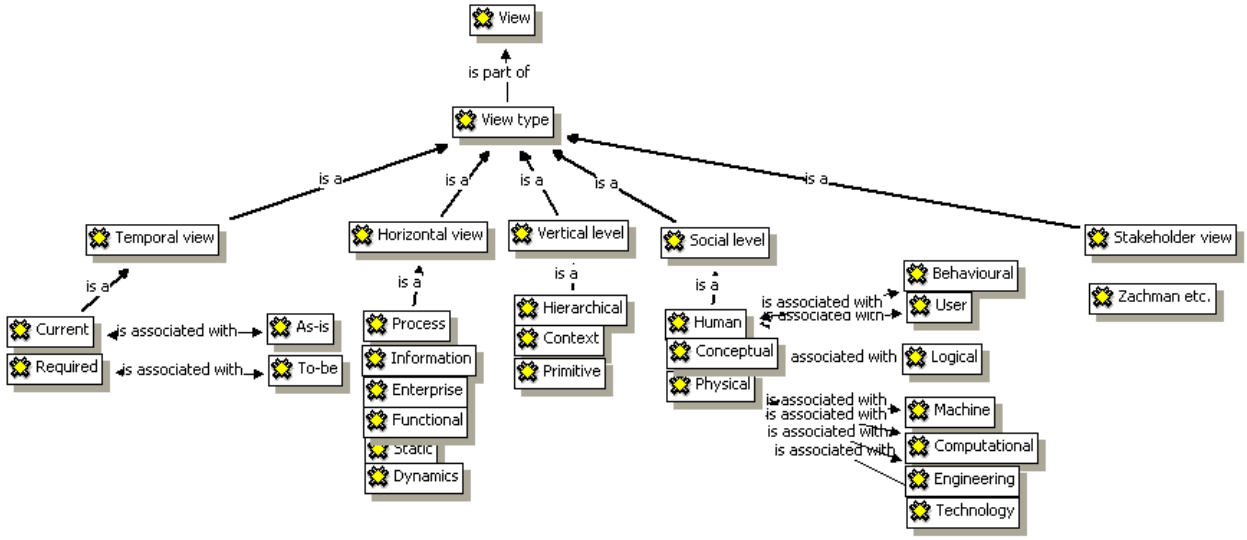


Figure 10-6: View-related codes

10.2.6.1 Link to techniques

The way in which *view* and its related codes are derived from the various modelling techniques is described in the following table:

Technique	Code
ALC/OLC	– Two perspectives are identified: a **process perspective** and a **behavioural perspective.**
AOR	– None on views.
ARM	– ARM provides a **structural perspective** of modelling.
BPMN	– BPMN provides a **human level view** of business processes vs. the web service-based execution languages that provide a **machine level view.**
DFD	– SSADM distinguishes the following types of DFD models: o **Current physical** – what system does o **Current logical** – how it does it o **Required logical** – what it should do o **Required physical** – how it should do it – The diagram describing the highest-level process is called a **context diagram.**
PERT and GANTT	– The same information can be viewed from many perspectives or **views,** e.g. Gantt, network (PERT), schedule.

Technique	Code
IDEF0	– IDEF0 provides a **functional** view. – A **context view** can be distinguished from the views on lower **levels.**
IDEF1	– IDEF1 provides an **information** view. – Models both **as-is** and **to-be**.
IDEF1X	– IDEF1X provides a **semantic** view.
IDEF3	– There can be more than one decomposition on a level, representing different **points of view.** – But there is one consolidated **objective view.**
IDEF5	– Contrasts **procedural** with **declarative** knowledge.
LAP	– Distinguishes the atomic, fibre and molecular **layers.**
ODP	– Identifies the following **perspectives** or **viewpoints:** o **Enterprise** – concerned with the business environment in which the system has to operate o **Information** – concerned with the information to be stored and processed by the system o **Computational** – concerned with a description of the system as a set of objects that interact at interfaces o **Engineering** – concerned with the mechanisms supporting system distribution o **Technology** – concerned with the detail of components from which the distributed system is constructed
RAD	– Provides a **dynamics view** of processes.
SQL	– Provides a **database** view.
SSM	– Provides a **soft systems** view.
UML	– Four views can be identified, each with a set of diagrams: o **Static view,** with class and object diagrams o **Dynamic view,** with sequence, collaboration and state chart diagrams o **Functional view,** with use case and activity diagrams o **Implementation view,** with package, component and deployment diagrams – **Packages** can be used to organise diagrams created during a project. That helps to focus on a topic or type of behaviour in the system.
Zachman	– The **perspectives** of or functions performed by main stakeholders: o **Planner** or scope – strategic in nature, on organisational boundary, and external environment is important o **Owner** or enterprise – all activities important to enterprise are described o **Designer** or system – logical level user requirements are specified o **Builder** or technology – more physical with some logical views, physical hardware specified o **Subcontractor** or components – detailed specifications o **Functioning system**

10.2.6.2 Link to theoretical foundations

No specific links.

10.2.7 Relationship

The code with the most related subcodes and which occurs in almost all techniques is *relationship*. It is called mostly *relationship* in the various techniques, but is also called *association*, *link* or *relation*. The main aspects of relationship derived from the grounded analysis are *properties* of relationships, and the *types* of relationships that can be identified. See Figure 10-7 for a summary of the main codes related to *relationship*, Figure 10-8 for a summary of the codes related to *relationships properties*, Figure 10-9 for a summary of the codes related to *general relationships*, Figure 10-10 for a summary of the codes related to **agent relationships** and Figure 10-11 for a summary of the codes related to *action relationships*.

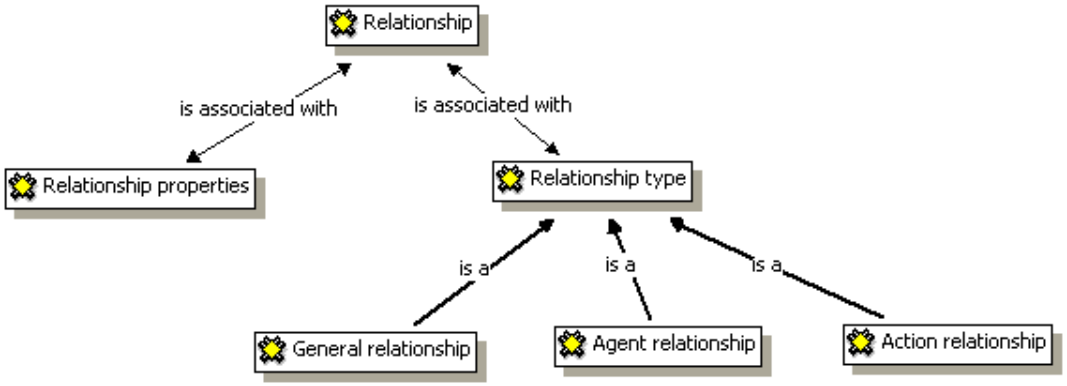


Figure 10-7: Relationship-related main codes

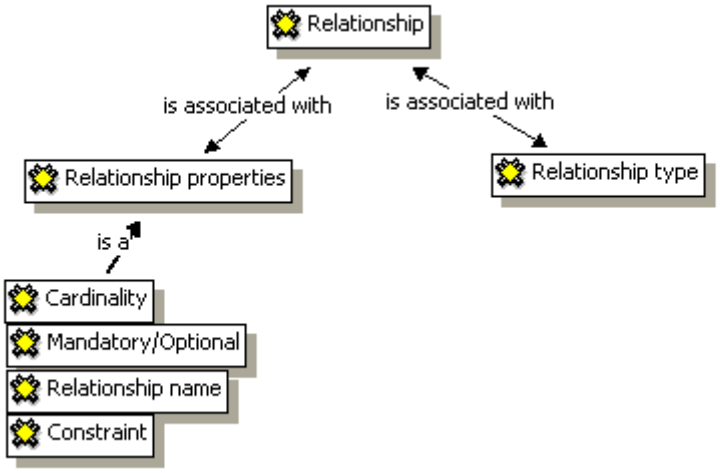


Figure 10-8: Relationship properties-related codes

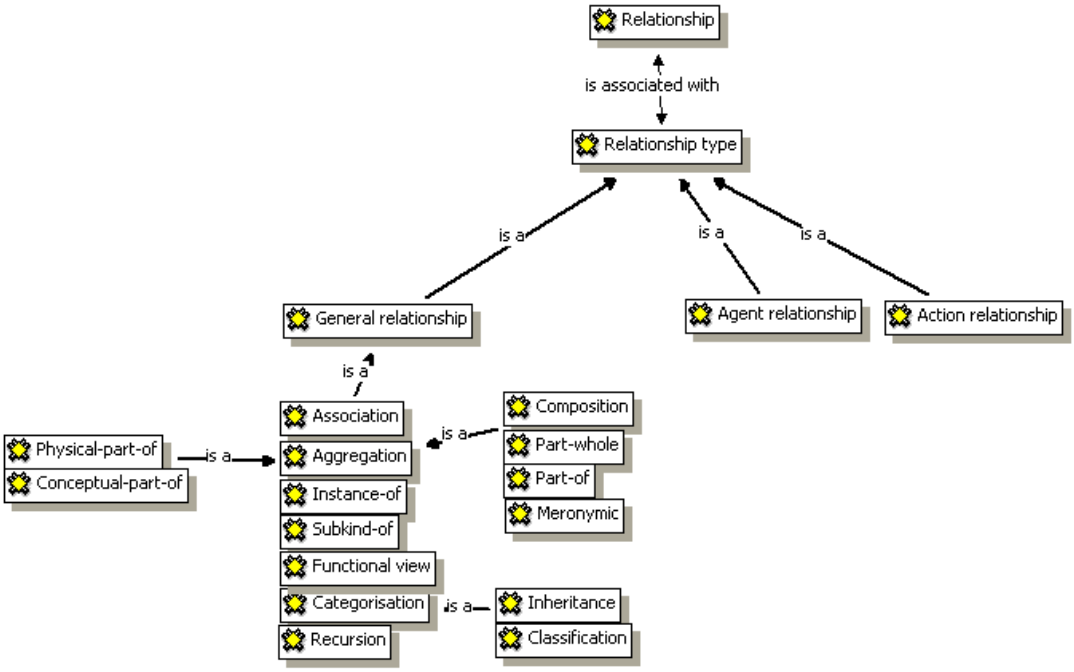


Figure 10-9: General relationship-related codes

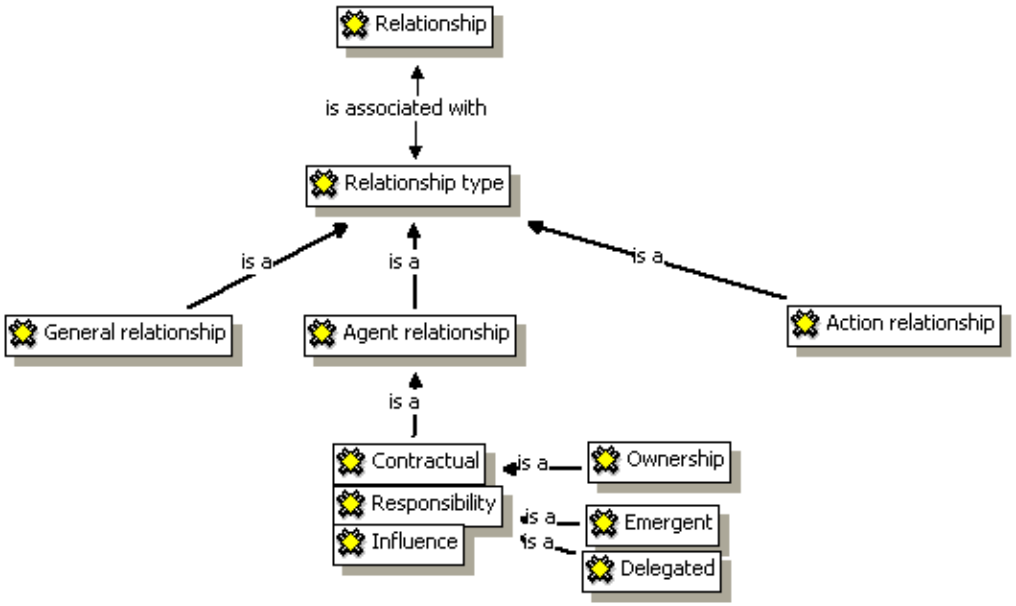


Figure 10-10: Agent relationship-related codes

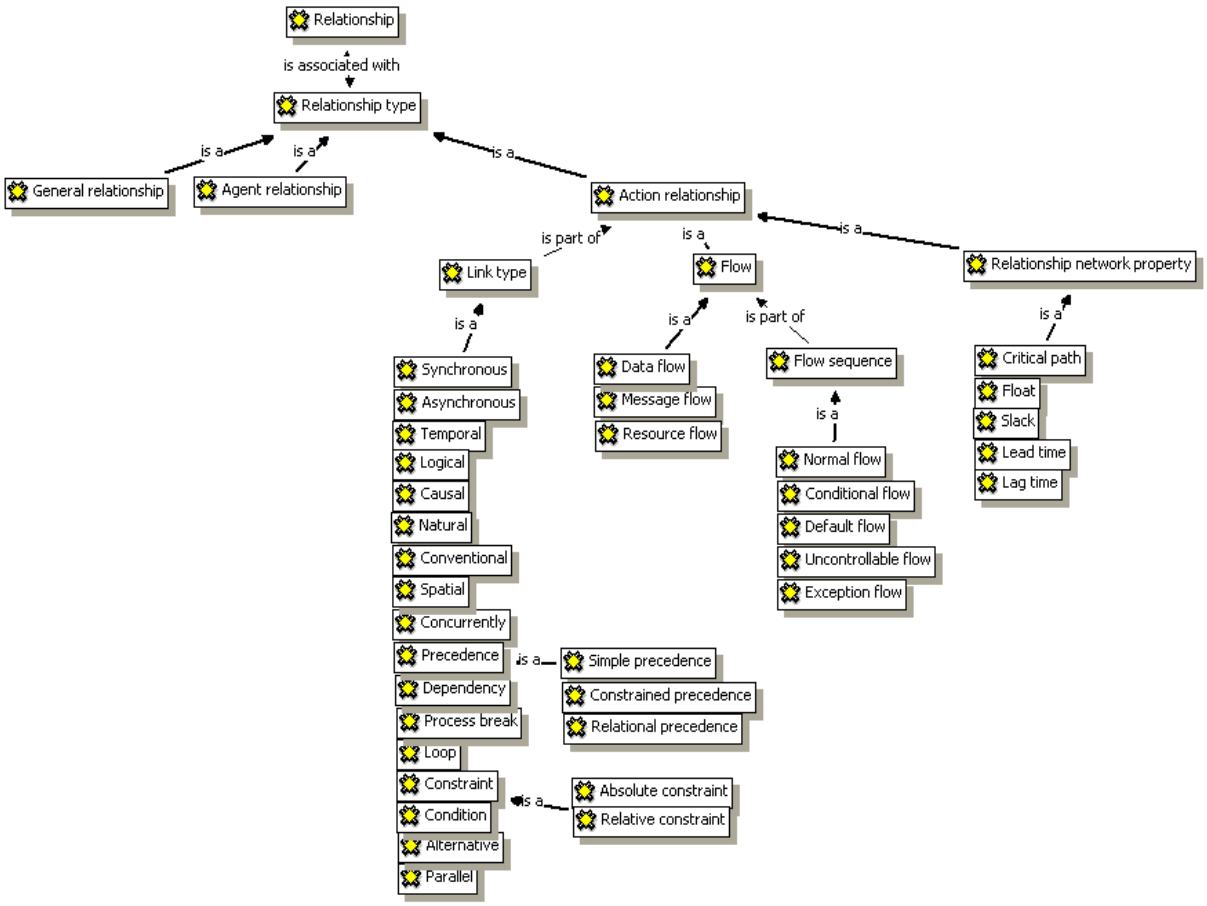


Figure 10-11: Action relationship-related codes

10.2.7.1 Link to techniques

The way in which *relationship* and its related codes are derived from the various modelling techniques is described in the following table:

Technique	Code
ALC/OLC	– Agents are related to other agents because of their **responsibility.**
AOR	– Entities participate in **relationships** with other entities. – Relationships (**associations**) are classified by means of **relationship types.** – There are two designated relationships between entity types that are independent of the application domain: **generalisation** (subclass, super class) and **composition** (component class).
ARM	– Agents have **contractual relationships** with other agents. – Agents can have **functional** or **ownership relationships** with objects. – Complex agents/objects are the **aggregation** of component agents/objects. – Relationships have **cardinality.** – **Responsibility** is a specific kind of relationship that agents/objects have with other entities. – **Emergent responsibility** is the responsibility that a complex agent has because it is considered to be a whole. – **Delegated responsibility** is when a complex agent/object might participate in some relationships only because its participants/components participate in it.
BPMN	– A **process** can consist of **subprocesses,** which can consist of **tasks.** – Activities can be connected by the following: o **Sequence flow,** which determines the order in which activities will be performed. o **Message flow** is the flow (send and receive) of messages between participants. o **Association** associates text or graphical information with flow object. – **Gateways** model control flow (decisions) in sequence flows (branching, forking, merging and joining of paths) and can either be the following: o **Exclusive gateways** that model “either/or” situations using XOR gateways o **Parallel gateways** that model parallel situations using AND gateways o **Inclusive gateways** that model “or” but not necessarily “either/or” situations using inclusive OR gateways o **Event-based gateways** that model decisions based on events

Technique	Code
DFD	– Analysis takes place in a **top-down** manner until processes can be described by a single active verb with a singular object.
PERT and GANTT	– Activities can be linked together by specifying **predecessor** activities. These links can be qualified further as follows: o **Start-to-finish** – when one task starts, the other can finish o **Start-to-start** – when one task starts, the other can start o **Finish-to-start** – when one task finishes, the other can start o **Finish-to-finish** – when one task finish, the other can finish – The links can also be qualified further by means of **lag** and **lead** times. – The activities can be organised hierarchically by means of a **work breakdown structure.**
IDEF0	– Model consists of a **hierarchical** series of diagrams.
IDEF1	– An **entity class** is a class of individual member entities. – The **relationship** between two entities has the following: o **Cardinality**: 1:m, 1:1, m:n, zero or one o A **label:** preposition-like words or verb-like words
IDEF1X	– Kinds of relationships are as follows: o **Identifying** and **non-identifying** o **Mandatory** and **optional** o **Parent** and **child** o **Categorisation** relationship (one entity is a type of another) – Relationships have 0, 1 or n **cardinality.** – Relationships are expressed as **verbs.**
IDEF3	– Processes can be linked together as follows: o **Simple precedence** – most common type, involves temporal precedence o **Constrained precedence** – defined by terms like must, ought and normative o **Relational** – e.g. one cannot approve one's own timesheet – **Link types** are temporal, logical, causal, natural and conventional. – Links involve **junctions**, paths that split or merge (diverge or converge), multiple parallel or alternative subprocesses. – These junctions can be **conjunctive** (AND), **disjunctive inclusive** (OR), **synchronous AND** (instances all start at the same time) or **synchronous OR** (instances all end at the same time).
IDEF5	– Individuals can be **complex** (consisting of many other **objects** of various kinds) or **simple.** – Kinds of relations: o **First and second order** ▪ **Relations** are also called **connections** or **associations.** Typically **binary** but can be **n–ary**. ▪ **Subkind** relation: kind-kind.

Technique	Code
	▪ **Instance-of** relation: kind-individual. ▪ **Is-a** relation: not used because it is ambiguous (one of the first two). ▪ **Part-of** relation: individual-complex object. o **Classification** ▪ **Functional inclusion:** e.g. hammer is a tool ▪ **State inclusion:** e.g. polio is a disease, hate is an emotion ▪ **Activity inclusion:** e.g. tennis is a sport, murder is a crime ▪ **Action inclusion:** e.g. lecturing is a form of talking, frying is a form of cooking ▪ **Perceptual inclusion:** e.g. a cat is a mammal, an apple is a fruit o **Meronymic – Physical part-of** ▪ **Component-of** – relates object and one of its components, e.g. wheel is part of bicycle ▪ **Stuff-of** – object is partly made of some material, e.g. bike is partly steel ▪ **Portion-of** – two similar objects, one included in the other, e.g. slice is part of pie o **Meronymic – Conceptual part-of** ▪ **Member-of** – Object is member of some collection. Objects do not have to be similar, except membership, e.g. cards are part of deck, tree is part of forest. ▪ **Activity-within** – Features or phases of activities, e.g. paying is part of shopping, dating is part of adolescence. o **Spatial relations** such as left-of, above, behind, inside, between, far, touching, beside, disjoint. o **Case relations** ▪ **Agent-instrument** – e.g. skier uses skis, soldier uses gun ▪ **Agent-object** – e.g. writer-paper, baker-flour ▪ **Action-recipient** – e.g. lie down-bed, type-keyboard ▪ **Action-instrument** – e.g. paint-brush, strum-guitar ▪ **Agent-action** – e.g. dog-bark, artist-paint o **Temporal relations** ▪ Two types: **time-interval** has beginning, end and duration attributes; and **time-point.** o **Dependency relations:** such as, depends-on, depends-on-causally, depends-on-existentially, existentially dependent, causally dependent. – A **relation** is a definite descriptor that refers to an association in the real world.
LAP	– A **transaction** is a sequence of coordination events.
ODP	– None
RAD	– Activities can be **decomposed.** – Roles consist of **component** activities.

Technique	Code
	– There is a **sequence** of activities (the logic).
SQL	– Has **control-of-flow** commands like, Begin … End. Break (exits loop), Return, Waitfor, While, Case, If … Else and Goto. – Provides for **recursion.**
SSM	– Systems have **subsystems.**
UML	– **Associations** identify interactions between actors and use cases. Each association becomes a **dialogue.** – Classes have **relationships** or association with each other, like simple, aggregate, composite, inheritance, qualified, reflexive. – **Composition** is used for aggregations where the lifespan of the part depends on the lifespan of the aggregate, e.g. book-chapter vs. team-player. – **Associations** define the following: o Participating classes o The association type o Name of association – verb or verb phrase o Direction o Multiplicity o Role, describes how object participates in association o Constraints, e.g. must have valid driver's licence o Association class, when data needs to be stored about association – Sequence and collaboration diagrams show the **interactions** between objects
Zachman	– Distinguishes **primitives** vs. **composites.**

10.2.7.2 Link to theoretical foundations

Chapter 4 on part-whole relationships is applicable to relationships. The following are the major insights gained from this chapter:

- Relationships are either topological (connections) or mereological (part-whole relationships (Wand et al. 1999).
- A part-whole relationship is where one thing (the whole) is the combination of two or more things (the parts).
- A major part of modelling involves decomposition – breaking down a system into its constituent parts.
- Wholes (also called *composites*) have properties. If these properties are in the parts, they are considered inherited properties, otherwise the properties are considered to be emergent (Wand et al. 1999).

- Parts can be attached/detached, arbitrarily demarcated, a temporal subset, a property, or connected/disconnected from the whole.
- Parts can be material or immaterial.
- Part-whole relationships have the following primary characteristics: idempotent, commutative, associative, at least one resultant (inherited) property, at least one emergent property, irreflexiveness at the instance level and antisymmetry (Opdahl, Henderson-Sellers and Barbier, 2001b). The following are some of the types of part-whole relationships identified: component/integral-object, member/collection, portion/mass, stuff/object, feature/activity, place/area (Gerstl and Pribbenow, 1995).

10.2.8 Language

This code *language* and related codes occur in many of the techniques. It is called mostly *language* in the various techniques. The main aspects of language derived from the grounded analysis are *language artefacts* and the *language layers*. See Figure 10-12 for a summary of the main codes related to language.

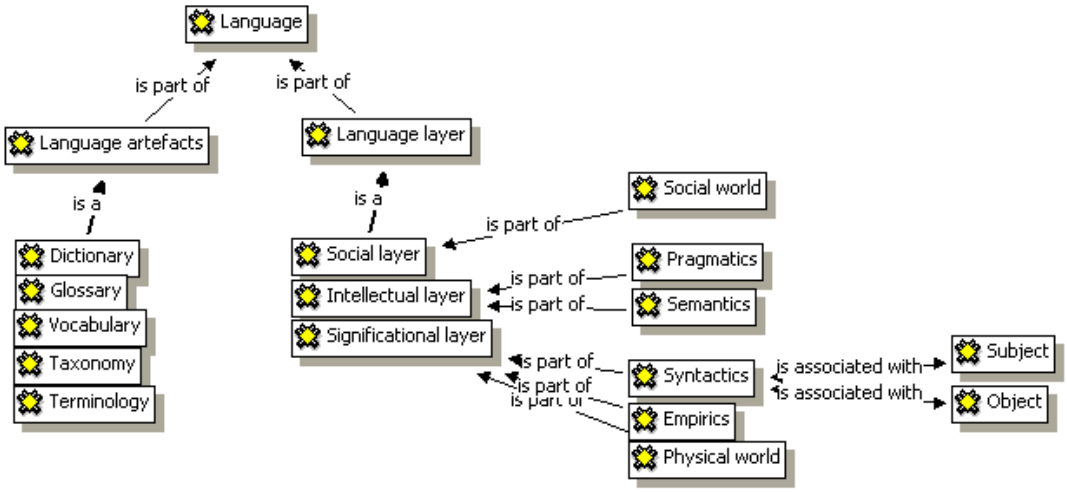


Figure 10-12: Language-related codes

10.2.8.1 Link to techniques

Technique	Code
ALC/OLC	– None
AOR	– None
ARM	– Consider the connection between **subjects** (agents) and **objects** or between subjects and subjects.
BPMN	– None
DFD	– None
PERT and GANTT	– None
IDEF0	– None
IDEF1	– Diagrams can be translated into **sentences.** – One modelling element is a **dictionary** (or **glossary**), describing the **meaning** of each modelling element.
IDEF1X	– Relationships are expressed as **verbs.**
IDEF3	– None
IDEF5	– In IDEF5, language sentences consist of the following: o **Constants** – words denoting objects o **Variables** – placeholders for constants o **Operators** – words and characters to form complex expressions
LAP	– The semiotic layers of communication are as follows: o Social layer (performa) relates to the social world o Intellectual layer (informa) relates to **pragmatics** and **semantics** o Significational layer (forma) relates to **syntactics, empirics** and the physical world
ODP	– None
RAD	– None
SQL	– None
SSM	– Non
UML	– State is typically an **adjective.**
Zachman	– None

10.2.8.2 Link to theoretical foundations

Many of the techniques discuss the layers or levels of linguistics (empirics, morphology, syntax, semantics and pragmatics) when defining ISD modelling, but none really take the concept to its full logical conclusion. The power of language is that a few basic codes (the alphabet, numbers and symbols) can be used to create an almost infinite number of words. In turn, these words, using a relatively small set of structuring rules, can be used to form an almost infinite number of phrases and

sentences, which can be understood by anybody who knows the specific language. All of these words and rules can be used to convey any meaning that a person could want to convey to other people.

Therefore, it became clear to the researcher that the most logical and flexible way to structure the grounded analysis codes into a coherent modelling technique is to structure them into a linguistic format. In other words, the central concept (axial code) in an analysis of ISD modelling is linguistics. Because the techniques did not really expand on the linguistic concepts they proposed, it was necessary to get most of the linguistic concepts from the theoretical foundations (see chapter 8 for an expansion of this).

10.2.9 Rule

The code *rule* and related codes occur in many of the techniques. It is called mostly *rule* in the various techniques. The main aspects of *rule* derived from the grounded analysis are *rule types*. See figure 10-13 for a summary of the main codes related to rules.

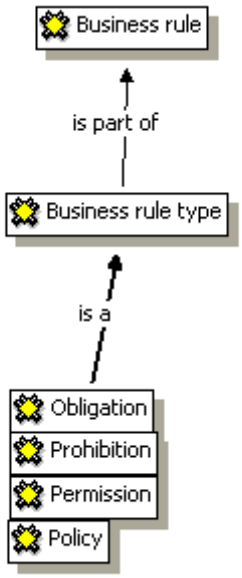


Figure 10-13: Rule-related codes

10.2.9.1 Link to techniques

Technique	Code
ALC/OLC	– None
AOR	– **Reaction rules** are used to specify the reactive and communicative behaviour of IS. – The typical format of a **reaction rule** is *ON event, IF condition, THEN action and effect*.
ARM	– None
BPMN	– None
DFD	– None
PERT and GANTT	– None
IDEF0	– **Control** are conditions required to produce correct output.
IDEF1	– **Dictionary** gives the meaning of each term.
IDEF1X	– Entity domains can have **domain rules.**
IDEF3	– Constrained precedence specify **constraints** like must, ought, within five minutes. – A constraint can be **relative** or **absolute.**
IDEF5	– None
LAP	– **Action rules** make a choice out of a set of possible acts and perform that act. – **Action rules** are considered atoms.
ODP	– **Structuring rules** are either obligations or policies. – An **obligation** is a prescription that a particular behaviour is required. – A **policy** is a set of rules related to a particular purpose. A rule can be expressed as an obligation, permission or prohibition.
RAD	– None
SQL	– **Constraints** are applicable to tables or columns. – **Integrity constraints** ensure that you can only reference existing records. – **Domain integrity** ensures that values entered comply with certain restrictions. – The **unique constraint** ensures that a record can only be entered once. – The **NOT NULL constraint** ensures that certain fields are always entered.
SSM	– None
UML	– **Guard conditions** restrict the use of activity transitions. – **Decisions** are either simple true/false situations or may involve a choice out of a number of options.
Zachman	– None

10.2.9.2 Link to theoretical foundations

In Chapter 3, business rules were discussed as a theoretical foundation. The Business Rules Group's classification has become the de facto classification of business rules (Hay and Healy, 2000):

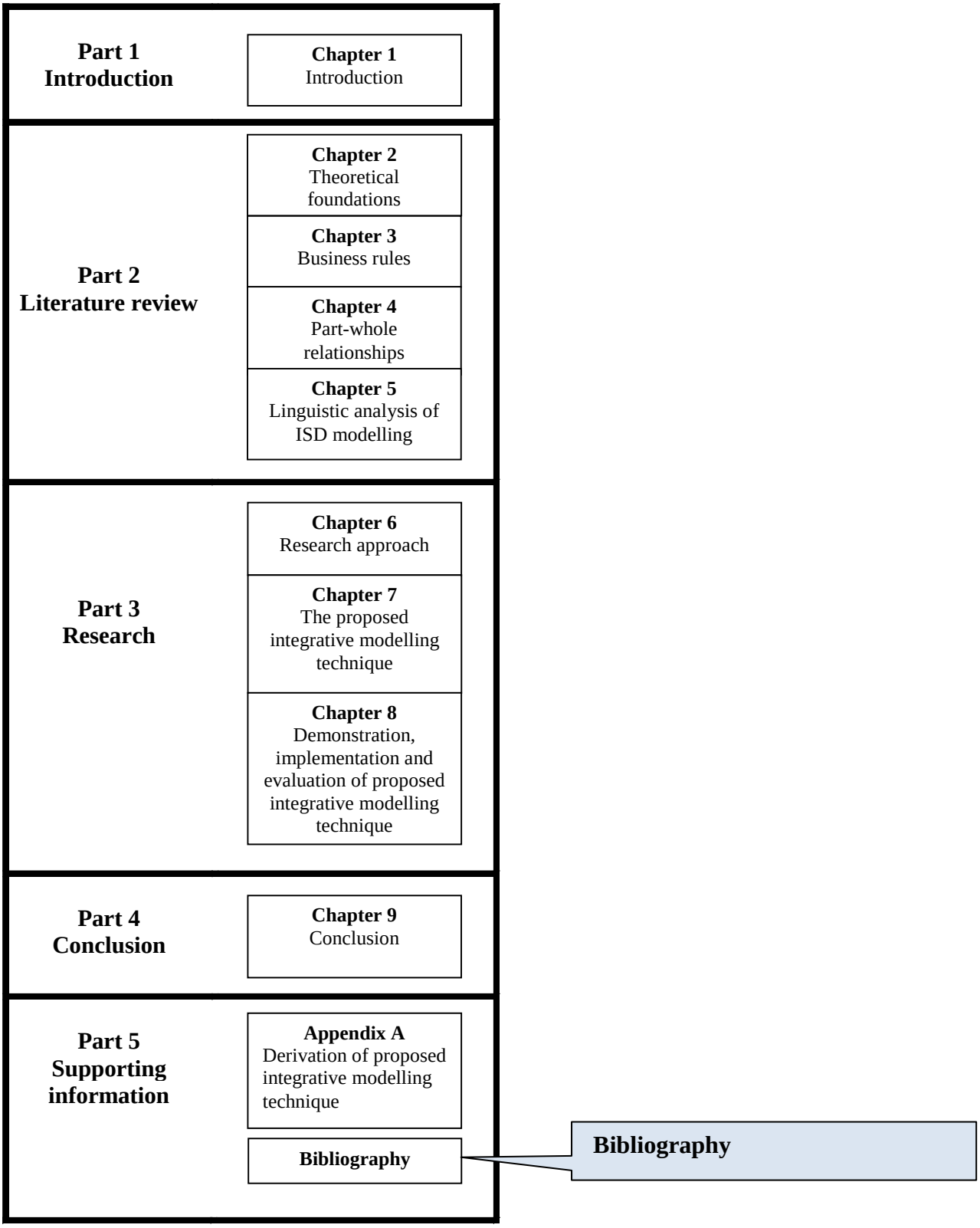
- *Terms*: defining a thing or data about it
- *Facts*: connections between terms
- *Constraints (*or *action assertions)*: allow or prohibit actions
- *Derivations* (or *inferences)*: the transformation of knowledge from one form to another

Business rules are related to business objects such as processes, activities, actors, goals and resources.

10.3 Conclusion

In this appendix, it was shown how the proposed integrative technique between business and ISD modelling was derived using grounded theory qualitative data analysis. The data used were extant ISD modelling techniques representative of the underlying structures and mechanisms (using critical realism terminology) that can empirically be evaluated, at least in terms of utility.

11. Bibliography



ADAMS, C. & AVISON, D. (2003) Dangers inherent in the use of techniques: Identifying framing influences. *Information Technology & People,* 16**,** 203–234.

AGUILAR-SAVEN, R.S. (2004) Business process modelling: Review and framework. *International Journal of Production Economics,* 90**,** 129–149.

ANAYA, V., BERIO, G., HARZALLAH, M., HEYMANS, P., MATULECICIUS, R., OPDAHL, A.L., PANETTO, H. & VERDECHO, M.J. (2010) The Unified Enterprise Modelling Language – Overview and further work. *Computers in Industry,* 61**,** 99–111.

ARTALE, A., FRANCONI, E., GUARINO, N. & PAZZI, L. (1996) Part-whole relations in object-centered systems: An overview. *Data & Knowledge Engineering,* 20**,** 347–383.

ATHENIKOS, S.J. & SONG, I-Y. (2008) A framework of ontology-based modeling patterns. *Proceedings of the Third International Conference on Design Science Research in Information Systems and Technology.* Atlanta, Georgia.

AVISON, D. & FITZGERALD, B. (2003) Where now for methodologies? *Communications of the ACM,* 46**,** 79–82.

AVISON, D. & FITZGERALD, G. (2006) *Information systems development methodologies, techniques & tools.* 4th Edition. McGraw-Hill.

BACKLUND, A. (2002) The concept of complexity in organisations and information systems. *Kybernetes,* 13, 31**,** 30–43.

BAJAJ, A., BATRA, D., HEVNER, A.R., PARSONS, J. & SIAU, K. (2005) Systems Analysis and design: Should we be researching what we teach? *Communications of the Association for Information Systems,* 15**,** 478–493.

BAJEC, M. & KRISPER, M. (2005) A methodology and tool support for managing business rules in organisations. *Information Systems,* 30**,** 423–443.

BAL, J. (1998) Process analysis tools for process improvement. *The TQM Magazine,* 10**,** 342–354.

BECKER, J. & NIEHAVES, B. (2007) Epistemological perspectives on IS research: A framework for analysing and systematizing epistemological assumptions. *Information Systems Journal,* 17**,** 197–214.

BENBYA, H. & MCKELVEY, B. (2006) Toward a complexity theory of information systems development. *Information Technology & People,* 19**,** 12–34.

BERNARD, T.J., POALINE III, E.A. & PARE, P-P. (2005) General systems theory and criminal justice. *Journal of Criminal Justice,* 33**,** 203–211.

BERNHAUER, J. (1996) Analysis of part-whole relation and subsumption in the medical domain. *Data & Knowledge Engineering,* 20**,** 405-415.

BHATTACHERJEE, A. & PREMKUMAR, G. (2004) Understanding changes in belief and attitude toward information technology usage: A theoretical model and longitudinal test. *MIS Quarterly,* 28**,** 229–254.

CAPUCHINO, A.M., JURISTO, N. & VAN DE RIET, R.P. (2000) Formal justi®cation in object-oriented modelling: A linguistic approach. *Data & Knowledge Engineering,* 2000**,** 25–47.

CARTER, M., LONG, C. & TRUEX, D.P. (2007) The notion of 'emergence' as a linguistic construct for eliciting security requirements in ISD. *Organizations and Society in Information Systems (OASIS), 2007 Workshop. IFIP 8.2.* Quebec, Canada.

CHANDLER, D. (2007) *Semiotics: The basics.* New York, Routledge.

CHARAF, M.C., ROSENKRANZ, C. & HOLTEN, R. (2010) Assessing language quality in the information systems development process – A theoretical

approach and its application. *International Conference on Information Systems (ICIS), 2010 Proceedings.* AIS Electronic Library (AISeL).

CHECKLAND, P. (1999) Systems thinking. IN CURRIE, W.L. & GALLIERS, B. (Eds.). *Rethinking management information systems.* Oxford University Press.

CHECKLAND, P. (2000) Soft systems methodology: A thirty year retrospective. *Systems Research and Behavioral Science,* 17**,** S11–S58.

CHEN-BURGER, J., ROBERTSON, D. & STADER, J. (2000) Formal support for an informal business modelling method, informatics research report. Division of Informatics, Artificial Intelligence Applications Institute, Institute for Representation and Reasoning, University of Edinburgh.

CHEN, P.P-S. (1976) The entity-relationship model – Toward a unified view of data. *ACM Transactions on Database Systems,* 1**,** 9–36.

COX, K. & PHALP, K.T. (2007) Practical experience of eliciting classes from use case descriptions. *The Journal of Systems and Software,* 80**,** 1286–1304.

DAENGBUPPHA, J., HEMMINGTON, N. & WILKES, K. (2006) Using grounded theory to model visitor experiences at heritage sites. *Qualitative Market Research: An International Journal,* 9**,** 367–388.

DAVISON, R.M. & MARTINSONS, M.G. (2011) Methodological practice and policy for organisationally and socially relevant IS research: An inclusive-exclusive perspective. *Journal of Information Technology,* 26.

DELUCA, D., GALLIVAN, M.J. & KOCK, N. (2008) Furthering information systems action research: A post-positivist synthesis of four dialectics. *Journal of the Association for Information Systems,* 9**,** 48–72.

DIAZ, I., LOSAVIO, F., MATTEO, A. & PASTOR, O. (2004) A specification pattern for use cases. *Information & Management,* 41**,** 961–975.

DIETZ, J.L.G. (2003) The atoms, molecules and fibers of organizations. *Data & Knowledge Engineering,* 47**,** 301–325.

DIK, S.C. (1997a) *The theory of functional grammar. Part 1. The structure of the clause.* Berlin, Mouton de Gruyter.

DIK, S.C. (1997b) *The theory of functional grammar. Part 2. Complex and derived constructions.* Berlin, Mouton de Gruyter.

DOBSON, P.J. (2002) Critical realism and information systems research: Why bother with philosophy? *Information Research,* 7.

ESCHENBACH, C. & HEYDRICH, W. (1995) Classical mereology and restricted domains. *International Journal of Human-Computer Studies,* 43**,** 723–740.

FRANCE, R., EVANS, A., LANO, K. & RUMPE, B. (1998) The UML as a formal modeling notation. *Computer Standards & Interfaces,* 19**,** 325–334.

FRANKEL, D.S., HARMON, P., MUKERJI, J., ODELL, J., OWEN, M., RIVITT, P., ROSEN, M. & SOLEY, R.M. (2003) The Zachman Framework and the OMG's model driven architecture. *Business Process Trends White Paper.*

FU, G., SHAO, J., EMBURY, S.M. & GRAY, W.A. (2004) Algorithms for analysing related constraint business rules. *Data & Knowledge Engineering,* 50**,** 215–240.

GAILLY, F. & POELS, G. (2007) Ontology-driven business modelling: Improving the conceptual representation of the REA ontology. IN AL., C.P.E. (Ed.). *ER 2007, LNCS.*

GEERTS, G. (2011) A design science research methodology and its application to accounting information systems research. *International Journal of Accounting Information Systems,* 12**,** 142–151.

GERSTL, P. & PRIBBENOW, S. (1995) Midwinters, end games and body parts: A classification of part-whole relations. *International Journal of Human-Computer Studies,* 43**,** 865–889.

GEYER, F. (1995) The challenge of sociocybernetics. *Kybernetes,* 24**,** 6–32.

GHANBARY, A. & DAY, J. (2009) Requirements modelling of business web applications: Challenges and solution. *Improving Systems and Software Engineering Conference (ISSEC).* Canberra, Australia.

GITT, W. (1997) *In the beginning was information.* Bielefeld, Christliche Literatur-Verbreitung e.V.

GLASER, B.G. (1978) *Theoretical sensitivity.* Mill Valley, CA, Sociology Press.

GLASER, B.G. (2003) *The grounded theory perspective II: Description's remodelling of grounded theory methodology.* Mill Valley, California, Sociology Press.

GLASER, B.G. & STRAUSS, A. (1967) *The discovery of grounded theory.* Chicago, Aldine Publishing Co.

GORDIJN, J. & AKKERMANS, H. (2002) Does e-business modeling really help? *The 36th Hawaii International Conference on System Sciences (HICSS'03). IEEE.* Hawaii.

GOULDING, C. (1998) Grounded theory the missing methodology on the interpretivist agenda. *Qualitative Market Research: An International Journal,* 1**,** 50–57.

GREGOR, S. (2006) The nature of theory in information systems. *MIS Quarterly,* 30**,** 611–642.

GREGOR, S. & JONES, D. (2007) The anatomy of a design theory. *Journal of the Association for Information Systems,* 8**,** 312–335.

GRUBER, T.R. (1995) Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies,* 43**,** 907–928.

GUIZZARDI, G. (2011) Ontological foundations for conceptual part-whole relations: The case of collectives and their parts. *International Conference on Advanced Information Systems Engineering.* London, UK.

HAMZA, H.S. & FAYAD, M.E. (2005) A novel approach for managing and reusing business rules in business architectures. IN HAMZA, H.S. & FAYAD, M.E. (Eds.). *AICCSA 2005 Workshop # 3 In Association with the 3rd ACS/IEEE International Conference on Computer Systems and Applications, AICCSA05.* Cairo, Egypt.

HASPELMATH, M. (2001) Word classes and parts of speech. *International Encyclopedia of the Social and Behavioral Sciences.* Elsevier Science Ltd.

HAY, D. & HEALY, K.A. (2000) *Defining business rules – What are they really*? The Business Rules Group.

HEATH, H. & COWLEY, S. (2004) Developing a grounded theory approach: A comparison of Glaser and Strauss. *International Journal of Nursing Studies* 41**,** 141–150.

HERBST, H. (1996) Business rules in systems analysis: A meta-model and repository system. *Information Systems,* 21**,** 147–166.

HEVNER, A.R. (2007) A three cycle view of design science research. *Scandinavian Journal of Information Systems,* 19**,** 87–92.

HEVNER, A.R. & MARCH, S.T. (2003) The information systems research cycle. *Computer*.

HEVNER, A.R., MARCH, S.T., PARK, J. & RAM, S. (2004) Design science in information systems research. *MIS Quarterly,* 28**,** 75–105.

HUGHES, J. & HOWCROFT, D.A. (2000) Grounded theory: Never knowingly understood. *Information Systems Review,* 4**,** 181–197.

IDEF0 (1993) *The Standard for Integration Definition for Function Modeling (IDEF0)*. Draft Federal Information Processing Standards Publication 183.

IIVARI, J. (2007) A paradigmatic analysis of information systems as a design science. *Scandinavian Journal of Information Systems,* 19**,** 39–64.

IIVARI, J., HIRSCHHEIM, R. & KLEIN, H. (2001a) A dynamic framework for classifying information systems development methodologies and approaches. *Journal of Management Information Systems,* 17**,** 179–218.

IIVARI, J., HIRSCHHEIM, R. & KLEIN, H.K. (2001b) Towards more professional information systems development: ISD as knowledge work. Global cooperation in the new millennium. *The 9th European Conference on Information Syststems.* Bled, Slovenia.

INTRONA, L.D. (1996) Notes on ateleological information systems development. *Information Technology and People,* 9**,** 20–30, 33.

JOHNSTON, S. (2004) Rational ® UML profile for business modeling. Rational software. IBM.

JOUBERT, P. (2009) Towards a linguistic analysis and representation of business rules. *The 1st International Workshop on Advanced Enterprise Repositories (AER 2009).* Milan, Italy.

KAPPES, S. (1997) Putting your IDEF0 model to work. *Business Process Management Journal,* 3**,** 151–161.

Kardasis and Loucopoulos (2004) i

KAYED, A. & COLOMB, R.M. (2005) Using BWW model to evaluate building ontologies in CGs formalism. *Information Systems,* 30**,** 379–398.

KILOV, H. & SACK, I. (2009) Mechanisms for communication between business and IT experts. *Computer Standards & Interfaces,* 31**,** 98–109.

KORNAI (2007) *Mathematical Linguistics*. Springer.

KROEZE, J.H. (2003) The semantic functions of embedded constructions in Biblical Hebrew. *Journal of NorthWest Semitic Languages,* 29**,** 107–120.

KROEZE, J.H. (2008) The so-called nominative uses of ta:e a semantic solution. *Journal for Semitics,* 17**,** 484–516.

LEPPANEN, M. (2006) Conceptual evaluation of methods for engineering situational ISD methods. *Software Process Improvement and Practice,* 11**,** 539–555.

LEWIS, R.T.V.N. (1994) From chaos to complexity: Implications for organizations. *Executive Development,* 7**,** 16–17.

LIANG, Y. (2003) From use cases to classes: A way of building object model with UML. *Information and Software Technology,* 45**,** 83–93.

LUUKKONEN, I., KORPELA, M. & MYKKANEN, J. (2010) Modeling approaches in the early phases of information systems development. *European Conference on Information Systems (ECIS), 2010.*

MAGEE, C. & DE WECK, O. (2004) *Complex system classification*. International Council on Systems Engineering (INCOSE).

MANSOUR, M. (2002) Systems theory and human science. *Annual Reviews in Control,* 26**,** 1–13.

MARKUS, M.L., MAJCHRZAK, A. & GASSER, L. (2002) A design theory for systems that support emergent knowledge processes. *MIS Quarterly,* 26**,** 179–212.

MARTIN, R.A., ROBERSTON, E.L. & SPRINGER, J.A. (2005) Architectural principles for enterprise frameworks: Guidance for interoperability. IN

Knowledge Sharing in the Integrated Enterprise. IFIP International Federation for Information Processing.

MARTINELLI, D.P. (2001) Systems hierarchies and management. *Systems Research and Behavioral Science,* 18**,** 69–82.

MAYR, H.C. & KOP, C. (2002) A user centered approach to requirements modeling. IN GLINZ, M. & MÜLLER-LUSCHNAT, G. (Eds.). *Modellierung 2002. Lecture Notes in Informatics.*

MAYR, H.C., KOP, C. & ESBERGER, D. (2007) Business process modeling and requirements modeling. *The First International Conference on the Digital Society (ICDS'07).*

MILLER, J.G. (1965) Living systems: structure and process. *Behaviorial Science,* 10 (4), 337-380.

MIHOUBI, H., SIMONET, A. & SIMONET, M. (1998) Towards a declarative approach for reusing domain ontologies. *Information Systems,* 23**,** 365–381.

MINGERS, J. (2004a) Critical realism and information systems: Brief responses to Monod and Klein. *Information and Organization,* 14**,** 145–153.

MINGERS, J. (2004b) Re-establishing the real: Critical realism and information systems. IN MINGERS, J. & WILLCOCKS, L. (Eds.). *Social theory and philosophy for informations systems.*

MINGERS, J. (2004c) Realizing information systems: Critical realism as an underpinning philosophy for information systems. *Information and Organization,* 14**,** 87–103.

MINGERS, J. (2008) Pluralism, realism, and truth: The keys to knowledge in information systems research. *International Journal of Information Technologies and the Systems Approach,* 1**,** 79–90.

MORA, M., OLMAN, O., FORGIONNE, G., PETKOV, D. & CANO, J. (2007) Integrating the fragmented pieces of IS research paradigms and frameworks: A systems approach. *Information Resources Management Journal,* 20**,** 1–22.

MYERS, M.D. (2009) *Qualitative research in business and management*. SAGE Publications Ltd.

NORAN, O. (2003) An analysis of the Zachman Framework for enterprise architecture from the GERAM perspective. *Annual Reviews in Control,* 27**,** 163–183.

NORAN, O. (2004) UML vs. IDEF: An ontology-oriented comparative study in view of business modelling. *International Conference on Enterprise Information Systems (ICEIS).* Port, Portugal.

ODEH, M. & KAMM, R. (2003) Bridging the gap between business models and system models. *Information and Software Technology,* 45**,** 1053–1060.

OEI, J.L.H., HEMMEN, L.J.G.T., VAN FALKENBERG, E.D. & BRINKKEMPER, S. (1992) The meta model hierarchy: A framework for information system concepts and techniques. IN FALKENBERG, E.D. (Ed.). *Technical Report.* Nijmegen, The Netherlands.

OMG (2007a) *Unified Modeling Language: Infrastructure. Version 2.1.1*.

OMG (2007b) *Unified Modeling Language: Superstructure. Version 2.1.1*.

OMG (2009) *Business Process Modeling Notation (BPMN). Version 1.2*.

OPDAHL, A.L., HENDERSON-SELLERS, B. & BARBIER, F. (2001a) Ontological analysis of whole-part relationships in OO-models. *Information and Software Technology,* 43**,** 387–399.

OPDAHL, A.L., HENDERSON-SELLERS, B. & BARBIER, F. (2001b) Ontological analysis of whole-part relationships in OO-models. *Information and Software Technology,* 43**,** 387–399.

ORLIKOWSKI, W.J. (1992) Duality of technology: Rethinking the concept of technology in organisations. *Organization Science,* 3**,** 398–427.

PARKER, L.D. & ROFFEY, B.H. (1997) Methodological themes: Back to the drawing board: Revisiting grounded theory and the everyday accountant's and manager's reality. *Accounting, Auditing & Accountability Journal,* 10**,** 212–247.

PATCHING, D. (1990) *Soft Systems Design.* London, UK, Pitman.

PEFFERS, K., TUUNANEN, T., ROTHENBERGER, M.A. & CHATTERJEE, S. (2008) A design science research methodology for IS research. *Journal of Management Information Systems,* 24**,** 45–77.

PERKINS, A. (2000) Business rules=meta-data. *34th International Conference on Technology of Object-Oriented Languages and Systems, 2000. TOOLS 34.*.

PMBOK (1996) *A guide to the Project Management Body of Knowledge*. PMI Publishing Division.

POO, D.C.C. (1999) Events in use cases as a basis for identifying and specifying classes and business rules. *Technology of Object-Oriented Languages and Systems.*

PURAO, S., BALDWIN, C.Y., HEVNER, A.R., STOREY, V.C., PRIES-HEJE, J., SMITH, B. & ZHU, Y. (2008) The sciences of design: Observations of an emerging field. Working paper 09-56.

RAM, S. & KHATRI, V. (2005) A comprehensive framework for modeling set-based business rules during conceptual database design. *Information Systems,* 30**,** 89–118.

RECKER, J. (2010) Opportunities and constraints: The current struggle with BPMN. *Business Process Management Journal,* 16**,** 181–201.

RECKER, J., INDULSKA, M., ROSEMANN, M. & GREEN, P. (2004) Do process modelling techniques get better? A comparative ontological analysis of BPMN. *16th Australasian Conference on Information Systems.* Sydney.

ROSCA, D. & D'ATTILIO, J. (2001) Business rules specification, enforcement and distribution for heterogeneous environments. *25th Annual International Computer Software and Applications Conference, (COMPSAC).* Chicago, USA.

RUSSELL, N., VAN DER AALST, W.M.P., TER HOFSTEDE, A.H.M. & WOHED, P. (2006) On the suitability of UML 2.0 activity diagrams for business process modelling. *The 3rd Asia-Pacific Conference on Conceptual Modelling (APCCM '06).* Darlinghurst, Australia.

SCHACH (2004) *Object oriented Analysis and design with UML and the unified process*. McGraw-Hill.

SEARLE, J.R. (1976) A classification of illocutionary acts. *Language in Society,* 5**,** 1–23.

SHAKER, P. (2010) Feature-oriented requirements modelling. *Software Engineering, 2010 ACM/IEEE 32nd International.* Waterloo, Ontario, Canada.

SHANKS, G., TANSLEY, E. & WEBER, R. (2004) Representing composites in conceptual modelling. *Communications of the ACM,* 47**,** 77–80.

SHEN, H., WALL, B., ZAREMBA, M., CHEN, Y. & BROWNE, J. (2004) Integration of business modelling methods for enterprise information system analysis and user requirements gathering. *Computers in Industry,* 54**,** 307–323.

SHINGHAL, R. (1992) *Formal concepts in artificial intelligence*. Chapman and Hall.

SINHA, A., PARADKAR, A., KUMANAN, P. & BOGURAEV, B. (2009) A linguistic analysis engine for natural language use case description and its application to dependability analysis in industrial use cases. *International Conference on Dependable Systems & Networks, 2009 (DSN '09), IEEE/IFIP*. Hawthorne, New York.

SOWA, J.F. & ZACHMAN, J.A. (1992) Extending and formalizing the framework for information systems architecture. *IBM Systems Journal,* 31**,** 590–616.

STABLER, E. (2010) *Introduction to Linguistics*, UCLA.

STEINKE, G. & NICKOLETTE, C. (2003) Business rules as the basis of an organization's information systems. *Industrial Management & Data Systems,* 103**,** 52–63.

STILLMAN, S. (2006) Grounded theory and grounded action: Rooted in Systems Theory. *World Futures,* 62**,** 498–504.

STRAUSS, A. & CORBIN, J. (1998) *Basics of Qualitative ResearchGrounded Theory Procedures and Technique (2nd Edition),* Newbury Park, London, Sage.

THE OPEN GROUP (2007) *The Open Group Architecture Framework (TOGAF) Version 8.1.1, Enterprise Edition*.

TOUSSAINT, P., BAKER, A. & GROENEWEGEN, L. (1997) Constructing an enterprise viewpoint - evaluation of four business modelling techniques. *Compter Methods and Programs in Biomedicine,* 55**,** 11–30.

TYNDALE-BISCOE, S., SIMS, O., WOOD, B. & SLUMAN, C. (2002) Business Modelling for Component Systems with UML. *The Sixth International ENTERPRISE DISTRIBUTED OBJECT COMPUTING Conference (EDOC'02).*

VAISHNAVI, V. & KUECHLER, W. (2004) Design Science Research in Information Systems. Last updated 30 September, 2011, last accessed 2 February 2012, http://desrist.org/desrist.

VALEIKA, L. & BUITKIENE, J. (2003) *An Introductory Course in Theoretical English Grammar,* Vilnius Pedagogical University.

VALIRIS, G. & GLYKAS, M. (2004) Business analysis metrics for business process redesign. *Business Process Management Journal,* 10**,** 445–480.

VARZI, A. (2010) Mereology. IN ZALTA, E. N. (Ed.) *The Stanford Encyclopedia of Philosophy (Spring 2011 Edition).*

VENABLE, J. R. (2006) The Role of Theory and Theorising in Design Science Research. *1st International Conference on Design Science – DESRIST* Claremont, CA.

VENABLE, J. R. (2010) Design Science Research Post Hevner et al: Criteria, Standards, Guidelines, and Expectations. *GLOBAL PERSPECTIVES ON DESIGN SCIENCE RESEARCH. Lecture Notes in Computer Science.* Springer.

VON BERTALANFFY, L. (1950) The Theory of Open Systems in Physics and Biology. *Science,* 111**,** 23–29.

VON HALLE, B. (2002) *Business Rules Applied: Business Better Systems Using the Business Rules Approach,* John Wiley and Sons Ltd.

WAGNER, G. (2002) The Agent–Object-Relationship metamodel: Towards a unified view of state and behavior. *Information Systems,* 28**,** 475–504.

WALLS, J., WIDMEYER, G. R. & EL SAWY, O. A. (1992) Design Theory for Vigilant EIS, Information Systems Research. *Information Systems Research,* 3**,** 36–59.

WAN-KADIR, W. M. N. & LOUCOPOULOS, P. (2004) Relating evolving business rules to software design. *Journal of Systems Architecture,* 50**,** 367–382.

WAND, Y. (1996) Ontology as a foundation for meta-modelling and method engineering. *Information and Software Technology,* 38**,** 281–287.

WAND, Y., STOREY, V. C. & WEBER, R. (1999) An Ontological Analysis of the Relationship Construct in Conceptual Modeling. *ACM Transactions on Database Systems,* 24**,** 494–528.

WAND, Y. & WANG, Y. (1996) Anchoring data quality dimensions in ontological foundations. *Communications of the ACM,* 39**,** 86–95.

WANG, T.-W. (2004) From General System Theory to Total Quality Management. *The Journal of American Academy of Business, Cambridge***,** 394–400.

WEIGAND, H. (1992) Assessing Functional Grammar for knowledge representation. *Data & Knowledge Engineering,* 8**,** 191–203.

WILCOX, P. A. & GURAU, C. (2003) Business modelling with UML: the implementation of CRM systems for online retailing. *Journal of Retailing and Consumer Services,* 10**,** 181–191.

WINTER, K., HAYES, I. J. & COLVIN, R. (2010) Integrating Requirements: The Behavior Tree Philosophy. *2010 8th International Conference on Software Engineering and Formal Methods (SEFM).* Brisbane, Australia.

WOLFSWINKEL, J. F., FURTMUELLER, E. & WILDEROM, P. M. (2012) Using grounded theory as a method for rigorously reviewing literature. *European Journal of Information Systems***,** 1–11.

WOODBURN, I. (1988) The Idea of 'System' and its use in 'hard' and 'soft' systems approaches. *Journal of Applied Systems Analysis,* 15**,** 49–54.

WYSSUSEK, B. (2006) On ontological foundations of conceptual modelling. *Scandinavian Journal of Information Systems,* 18**,** 63–80.

XIA, W. & LEE, G. (2005) Complexity if ISD projects: Conceptualisation and measurement development. *Journal of Management Information Systems,* 22**,** 45–83.

YUE, T., BRIAND, L. C. & LABICHE, Y. (2009) A Use Case Modeling Approach to Facilitate the Transition Towards Analysis Models: Concepts and Empirical Evaluation. *Models 2009.*

ZACHMAN, J. A. (1987) A Framework for information systems architecture. *IBM Systems Journal,* 26**,** 276–292.

ZACHMAN, J. A. (1999) A framewrk for information systems architecture. *IBM Systems Journal,* 38**,** 454–470.

ZUR MUEHLEN, M. & INDULSKA, M. (2010) Modeling languages for business processes and business rules: A representational analysis. *Information Systems,* 35**,** 39, 379–390.