



NTNU – Trondheim
Norwegian University of
Science and Technology

The Molecular Modeling Language GEMAL

Alexander Andersson

Master of Science in Computer Science

Submission date: Januar 2015

Supervisor: John Krogstie, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Sammendrag

I denne oppgaven brukes en referansem modell, en ontologi og et vitenskaplig grunnlag laget for konstruksjon av visuelle notasjoner til å studere det molekylære modelleringspråket GEMAL. Oppgaven følger en vitenskaplig forskningsmodell med fokus på design. Et språkforslag for GEMAL, laget av veilederen for denne oppgaven, John Krogstie, dannet utgangspunktet for videre design og utvikling av språket. Modelleringspråket har blitt vellykket implementert i en applikasjon for design og bruk av modelleringspråk. Case-studier, en nettbasert spørreundersøkelse og ekspertevalueringer har blitt benyttet, med mål om å kunne vise hvorvidt et molekylært modelleringspråk kan være godt utrustet til å takle mange av de utfordringene som stilles til modelleringspråk i feltet for konseptuell modellering i dag.

Resultater indikerer at GEMAL bringer noen interessante ideer til feltet. Ved å bruke implementasjonen av modelleringspråket har det blitt vist at det er mulig å representere en rekke tradisjonelle modellering-caser i språket, og at modeller som stammer fra ulike modelleringspråk tilhørende ulike modelleringsperspektiver kan reproduseres ved hjelp av GEMAL.

Abstract

This project follows the design science research methodology, and uses a reference model, an ontology, and a scientific basis for constructing visual notations, to study a molecular modeling language (GEMAL). The project takes a language proposal for the modeling language, created by the project's supervisor John Krogstie, as a starting point for further design and development. The modeling language has been successfully implemented in a software application for modeling languages. Case studies, an online survey, and an expert evaluation have been conducted in an effort of uncovering whether a molecular modeling language like GEMAL is better equipped to tackle some of the many challenges posed to modeling languages in the conceptual modeling field today.

Results indicate that GEMAL brings some interesting ideas to the field. Using the implementation of the modeling language, it has been shown that GEMAL is able to represent a range of traditional modeling cases, and that it can reproduce models stemming from different modeling languages that adhere to different modeling perspectives.

Problem Description

Traditional modelling languages focuses on modeling according to a specific modeling perspective, e.g. being primarily process-oriented or object-oriented. Even if newer languages such as UML and EEML combines several perspectives, they primarily do this by combining several single-perspective modelling languages. One result of this is that the resulting languages are large and cumbersome to learn and to use.

An alternative approach is so-called holistic modelling languages, where a limited number of concepts can be combined in an orthogonal fashion. The task is to work further on one proposal for such a language (GEMAL), implement this in the METIS modelling tool, and evaluate this through case studies in different fields.

Supervisor: John Krogstie

Preface

The project has been defined in consultation with my supervisor, Professor John Krogstie at the Department of Computer and Information Science (IDI) at the Norwegian University of Science and Technology (NTNU). I would like to thank him for his guidance and support throughout the project. I started fresh on this project in the fall of 2014, the preceding specialisation project was on a different topic.

Also, a big thanks to the experts on the topic of conceptual modeling that participated in the expert evaluation of the modeling language; their feedback was inspiring and valuable.

I would like to extend my thanks and appreciation to those who participated in the project's online survey.

Last, but not least, I wish to thank my family and friends for their continuous support and encouragement throughout my education.

Trondheim, 30.01.2015

Alexander Andersson

Contents

Sammendrag	i
Abstract	ii
Problem Description	iii
Preface	iv
1 Introduction	2
1.1 Motivation	2
1.2 Problem Definition	2
1.3 Project Description	3
1.4 Report Outline	4
2 Background	5
2.1 Conceptual Modeling	5
2.1.1 Historical Background	5
2.1.2 Modeling Perspectives	6
2.1.2.1 Behavioural	6
2.1.2.2 Functional	7
2.1.2.3 Structural	8
2.1.2.4 Goal and Rule	8
2.1.2.5 Object Perspective	8
2.1.2.6 Communication	9
2.1.2.7 Actor and Role	9
2.1.2.8 Topological	10
2.1.2.9 Other Views on modeling Perspectives	10
2.2 Enterprise Modeling	11
2.2.1 Enterprise Architecture Frameworks	11
2.2.1.1 The Zachman Framework	11
2.2.1.2 ArchiMate	12
2.2.1.3 Other Frameworks	12
2.3 Weaknesses of Current Approaches to Conceptual Modeling for Enterprises	13

2.3.1	Holistic modeling	14
2.3.2	Holistic modeling in Context	16
2.3.2.1	Facet modeling	16
2.3.2.2	Integrated use of Enterprise and IS Models	17
2.3.2.3	Multi-Perspective Methodologies	18
2.3.2.4	Example Notation: SeeMe	19
2.4	Evaluation of Modeling Languages	20
2.4.1	Feature Comparison	21
2.4.2	Theoretical and Conceptual Investigation	21
2.4.2.1	Establishing a Theoretical Basis	21
2.4.2.2	Metamodeling and Semiotics	22
2.4.2.3	Metrics Analysis	22
2.4.2.4	Paradigmatic Analysis	23
2.4.2.5	Contingency Identification	23
2.4.2.6	Ontological Evaluation	23
2.4.2.7	Cognitive Evaluation	24
2.4.3	Empirical Evaluation Techniques	24
2.4.3.1	Surveys	24
3	Research Design and Methodology	26
3.1	Conceptual Framework	26
3.1.1	Research Questions	26
3.2	General Research Approach	27
3.2.1	The Design Science Research Process	27
3.2.2	Designing for modeling Language Quality	29
3.2.2.1	The SEQUAL Framework	29
3.2.2.2	Moody's Notation: a Scientific Basis for Constructing Visual Notations	32
3.3	Data Collection	34
4	Initial Overview of GEMAL	35
4.1	Language Proposal as a Starting Point	35
4.1.1	Basic Concepts	35
4.1.2	Language mechanisms	36
4.2	Initial Evaluation of GEMAL's Language Quality	38
4.2.1	Domain Appropriateness	38
4.2.2	Comprehensibility Appropriateness	43
4.2.3	Participant Appropriateness	46

4.2.4	Tool Appropriateness	46
4.3	Summary	47
5	Implementation of GEMAL	48
5.1	Expanding the Language	48
5.1.1	Metamodel	48
5.1.2	Things	49
5.1.3	State Space of Things	49
5.1.4	Modality Space of Things	51
5.1.5	Introducing Complexity and Vagueness	52
5.1.6	Concretised Relationships	53
5.1.6.1	Relationship Visualisation Sceme	54
5.1.7	Logical Gates	54
5.1.8	Instantiation Level	55
5.1.9	Expressing Events in GEMAL	56
5.1.10	Properties of Things	56
5.1.11	Changes to Symbols	59
5.1.12	View Styles: Optional Increase to Visual Expressiveness	59
5.1.13	Other Additions to GEMAL	61
5.1.13.1	Layout Strategies	61
5.1.13.2	Other functionality	61
6	Case Studies	63
6.1	Case 1: Comparing GEMAL to Data Flow Diagram	63
6.2	Case 2: Comparing GEMAL to Entity Relationship	64
6.3	Case 3: Comparing GEMAL to i*	66
6.4	Case 4: Comparing GEMAL to ArchiMate	68
6.5	Case 5: Comparing GEMAL to Unified Modeling Language	68
6.6	Case 6: Arranging a Conference (Large-Scale GEMAL Model)	71
7	Final Evaluation	73
7.1	Semiotic Survey	73
7.1.1	Survey Results	74
7.1.2	Analysis of Results	75
7.2	Expert Evaluation	76
7.2.1	Results	77
7.2.2	Summary and Analysis of Results	83

8 Discussion	85
8.1 Viability of GEMAL in its current State	85
8.2 Recommended Design Guidelines for GEMAL models	85
8.3 Limitations	86
9 Conclusion and Further Work	87
9.1 Conclusion	87
9.2 Further Work	88
A Acronyms	89
B Implementation Details	92
B.1 Metamodel Structure	92
B.2 Elements in a Metamodel	92
B.3 Files in the GEMAL Metamodel	94
B.4 Template Files	95
C Case Study Descriptions	96
C.1 Order Management System	96
C.2 Arrangement of International Conferences and Events	97
C.2.1 Overall Setting (Basis for Type-Level Model)	97
C.2.2 Basis for Instance-Level Model (Most Information Found on the Web)	99
D Online Survey	100
E Survey Comments	103
F Expert Evaluation Questions	106
G GEMAL Language Overview	110
H Digital Attachments	113
Bibliography	114

List of Figures

2.1	Examples of early leading conceptual modeling approaches	6
2.2	Core BPMN 2.0 events	7
2.3	View model of the ArchiMate Architectural Framework	13
2.4	An example Facet Item (Facet modeling)	17
2.5	Relationship between goals and tasks in EEML	18
2.6	Overview of UML diagrams in UML 2.4.1	19
2.7	Visualisation of concepts in the SeeMe notation	19
3.1	The design science research process	28
3.2	Illustration of the SEQUAL framework	31
4.1	Symbols used for different specialisations of thing in the GEMAL language proposal	36
4.2	Examples of composite concepts from the GEMAL language proposal	37
4.3	An example of a high level process from the GEMAL language proposal	38
5.1	GEMAL Metamodel	49
5.2	GEMAL State Transitions	51
5.3	Updated visualisation in the GEMAL implementation	54
5.4	GEMAL logical gates	55
5.5	Visualisation of an example GEMAL Person	58
5.6	Updated symbol for GEMALs location concept	59
5.7	Example of the difference between basic and advanced view style for GEMAL things	60
6.1	DFD model and a corresponding GEMAL version	64
6.2	ER model and a corresponding GEMAL version	65
6.3	A closer inspection of the GEMAL model in fig. 6.2b	65
6.4	i* model and a corresponding GEMAL version	67
6.5	ArchiMate model and a corresponding GEMAL version	68
6.6	Original UML Diagrams used in case 5	70
6.7	GEMAL version of the UML Diagrams in fig. 6.6	70

6.8	Figures illustrating the 'Arranging a Conference' case	72
B.1	Structure of GEMAL metamodels, as they appear in Metis	94
B.2	The structured GEMAL template	95

List of Tables

- 3.1 Design science guidelines 28
- 4.1 Ontological matches between GEMAL and the BWW-model 39
- 7.1 Results for online survey question 1 74
- 7.2 Results for online survey question 2 75
- E.1 Comments stated by respondents of the GEMAL online survey 103

Chapter 1

Introduction

1.1 Motivation

Conceptual modeling approaches within the information systems field as we know it today were introduced on a large scale around 30 years ago, with the developments and commissioning of techniques including DFDs [19] and ER diagrams [16].

Since then, work in the field has shifted from trying to develop the perfect conceptual modeling language that captures *all* the important concepts of the world, to developing languages that are tailored for specific tasks by adhering to modeling perspectives or structuring principles. The given perspective will often prescribe how different aspects of the language are promoted, represented, dedicated to, visualised and sequenced relative to others [58]. As such, the choice of modeling language can be a huge limiting factor on how the modeller can express environments, knowledge, processes and more.

Newer languages such as UML and EEML combine several perspectives in attempt to overcome this weakness. However, the resulting languages have proven to be large and cumbersome to learn and to use [69].

The aim of this study is to explore the usefulness and expressiveness of a modeling language that avoids enforcing any modeling perspective.

1.2 Problem Definition

The goal of this project is to extend and realise a molecular enterprise modeling language ('General Enterprise Modelling and Activation Language', hereafter abbreviated as 'GEMAL'), and to evaluate its usefulness through case studies in different fields, assessments in accordance with quality frameworks, and empirical evaluations.

The multi-perspective modeling language strives to provide a modeling environment that

is better able to capture and express the concepts that are of value to the various stakeholders involved in conceptual modeling tasks, while retraining some of the structure and clarity often found in traditional modeling languages.

1.3 Project Description

Contributions of this project include the realisation, implementation and evaluation of the modeling language GEMAL. GEMAL was first proposed by John Krogstie in his 2012 book [45]. In this project, the language has been implemented in a well known modeling tool.

This study follows the guidelines for design science research. The main artifact produced is an implementation of GEMAL in a modeling tool with sample cases and suggested startup material. Project outcomes also include a thorough evaluation of GEMAL, along with suggestions for further improvements of the modeling language.

The project followed a design and development centered approach. As such, work started in the design & development process with regard to the GEMAL implementation. However, since the work done on GEMAL has been based on a language proposal by the original author, demonstration, evaluation, and communication of previous work became a necessary step in this first process. This way the problem to be solved was identified and motivated, and the proposed objectives of a solution were outlined.

The implementation of GEMAL was evaluated through a literature review, case studies, a online survey, and expert evaluations. Results were used to guide work as the project processed. Also, results helped shape usage guidelines for the language, and helped outline recommendations for future work on GEMAL. Results are presented in this report.

Note that the project is not based on any preceding projects carried out by the author. The author's specialisation project dealt with a vastly different set of topics than this project, and as such it is not used as a basis in this thesis.

1.4 Report Outline

The rest of the report is structured as follows:

Chapter 2 Background presents the main findings of the literature study conducted in this project.

Chapter 3 Research Design and Methodology describes the research questions and the research model with its constructs. The hypotheses are presented before the chapter continues with a presentation of the methodological choices and the research design. The validity and reliability of the measures are discussed and, finally, the analysis methods are introduced.

Chapter 4 Initial Overview of GEMAL provides an initial overview of GEMAL and the state the language was in before it was implemented in this project. The chapter continues with a theoretical and conceptual investigation of GEMAL, where the modeling language is measured against conceptual frameworks and notations. This investigation laid the foundation for expanding on the language as a part of this project.

Chapter 5 Implementation of GEMAL gives an overview of changes made to GEMAL as it was implemented in the Metis modeling tools. Some changes were made to accommodate weaknesses found in chapter 4, and others because of opportunities provided in the Metis metamodeling software.

Chapter 6 Case Studies presents case studies that were developed in order to test the expressiveness of GEMAL. The studies are based on descriptions that are listed in the appendix. The listed cases were utilised in expert evaluations.

Chapter 7 Final Evaluation documents the evaluation effort that was carried out towards the end of the project, after GEMAL had been implemented and tested in case studies. Results of the online survey and the expert evaluation are presented.

Chapter 8 Discussion looks at project results, and compares them to research questions and goals, to provide an indication of the strong and weak points of this study.

Chapter 9 Conclusion and Further Work concludes the work of this project, and outlines a suggested direction for further work on GEMAL and related material.

Chapter 2

Background

2.1 Conceptual Modeling

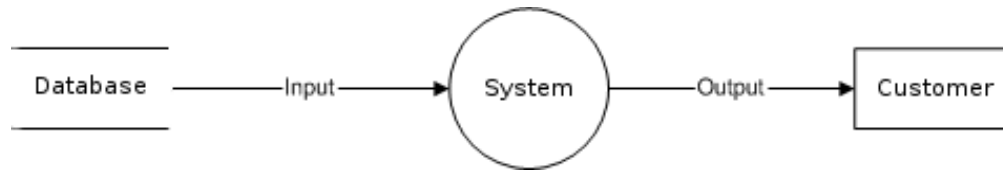
2.1.1 Historical Background

Conceptual modeling involves building a representation of selected phenomena in some domain [47, 81]. Research in the field can be tracked back to the 1970s, starting out as primarily focused on developing modeling techniques (e.g., [90, 16, 19]). Early modeling approaches included ER-diagrams and DFD [19, 16], see figure 2.1. Research followed the developments and adoptions of database systems and systems analysis techniques - the focus was mainly on finding *the* right modeling notation for any situation [45], with every new approach supposedly trumping the rest. Researchers became frustrated with the situation, depicted by the adoption of terms like "YAMA" ("Yet another modeling approach") [54] and "NAMA" ("Not another modeling approach") [31].

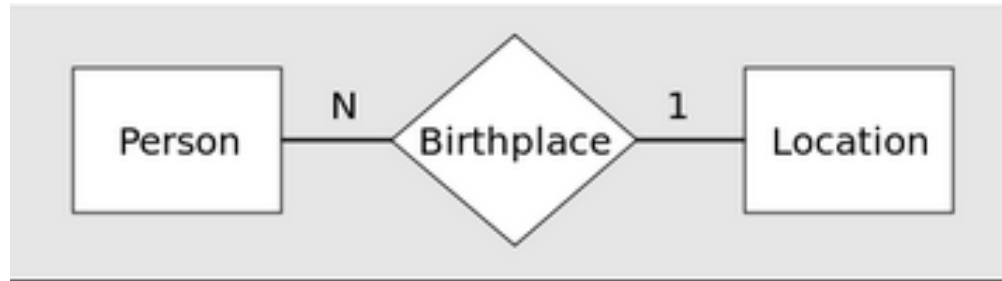
By the late 1980s, it was to a greater degree understood that requirements of a modeling language depend on the objectives of modeling. Meta-modeling approaches started appearing in early 1990s (i.e. MetaEdit+ [42]), allowing projects and organisations to extend existing modeling languages and notations, or even to create new languages. Research interest in the field was somewhat low in the 1990s, but renewed interest occurred in the light of the rising popularity of the object-oriented approach [81]. Conceptual modeling was also paired with requirements eliciting in B2B and e-commerce systems, data and process models for enterprise resource planning systems, business process reengineering, and software reuse specifications. It was thought to be usable for understanding the broader social context of system development. [81].

¹Figure source: "DataFlowDiagram Example" by Ilario licensed under CC BY-SA 3.0

²Figure source: Adapted from "ERD Representation" by Benthompson licensed under public domain.



(a) A simple Data flow diagram (DFD) example using Yourdon/Demarco notation [19]¹



(b) A simple Entity Relationship (ER) diagram using Chen notation [16]²

Figure 2.1: Examples of early leading conceptual modeling approaches

2.1.2 Modeling Perspectives

This section provides a brief overview of modeling perspectives used in different traditional conceptual modeling languages, seeking to outline some of the most influential work on conceptual modeling over the years. In many cases it will be difficult to classify a specific approach solely according to one perspective within a scheme. The following is based on the eight perspectives outlined by Krogstie [45]. In section 2.1.2.9 we will compare this perspective distinction to other relevant distinctions. We will briefly outline a selection of modeling languages together with the perspective they can be grouped under. These specific modeling languages will be referred to in later parts of this thesis.

2.1.2.1 Behavioural

The main phenomena of most languages adhering to this perspective are states and transitions between states. State transitions are triggered by events. Events may also have conditions and actions. An early example from the 1960s on which much later work has been based, are Petri nets. Other examples include Statecharts, state transition diagrams, state transition matrices and system dynamic models.

2.1.2.2 Functional

The main phenomena class in this perspective is the transformation. A transformation can be an activity that transforms a set of phenomena to another (possibly empty) set. Data flow diagram (DFD) is a classical example of a modeling language following this perspective (see fig. 2.1a for a simple example). It uses the concepts process, store, flow and external entity to describe a situation. With its dependency on external stimuli, DFDs are best used to model reactive systems. Other examples are EPC, BPMN and UML activity diagrams.

Example Language: BPMN

Business Process Model and Notation (BPMN) is a standard for business process modeling that provides a graphical notation for specifying business processes in a Business Process Diagram (BPD) [71]. The flow chart technique used is similar to activity diagrams from UML. BPMN strives to provide a standard notation that is readily understandable by all business stakeholders, including business analysts, technical developers and business managers. The four basic elements of BPMN are flow objects (events, activities, gateways), connecting objects (sequence flow, message flow, association), swim lanes (pools, lanes) and artefacts (data objects, groups, annotations). Bruce Silver has written an implementer's guide for BPMN, that provides more detail on the notation [70].

Since events are a core part of BPMN, and to be considered for GEMAL in a later chapter, we have included the core event types of BPMN in figure 2.2 for reference. BPMN 2.0 has 63 event types in total.















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

Figure 2.2: Core BPMN 2.0 events³. **Source:** Adapted from [70].

³B. Silver claims that these types comprise the most common usages of the "Big 3" types of BPMN 2.0 events (timer, message, error) in a small and readily learnable subset. His book covers the full set of events [70].

2.1.2.3 Structural

Approaches following this perspective are generally focused on describing the static structure of a system, with the main construct being the 'entity' - sometimes called object, concept, thing or phenomena. ER diagrams comprised the first structural modeling languages to have a big impact on the conceptual modeling world (see a simple example in fig. 2.1b). ER components include entities, relationships and attributes. Other examples of languages include GSM, NIAM and ORM. Semantic networks are structural modeling languages that have seen much use in the AI world (i.e. the WordNet language [50]).

2.1.2.4 Goal and Rule

Languages of this perspective represent knowledge in terms of goals and rules. Alethic (rules of necessity) and deontic (socially constructed) rules can influence the actions of a set of actors. Goals and rules were important tools in the expert systems of the 1980s, and have in more recent times seen use in rule-based systems, and goal oriented modeling in the field of requirement specification. Goals and rules are combined with structural modeling elements in ontologies (i.e. the Web Ontology Language (OWL [33])). Approaches such as Tempora (see the closely related languages ERT, PID and ERL) link rules to other models [48]. In goal and rule based modeling there has to be made a tradeoff between expressiveness and efficiency of (automatic) reasoning.

2.1.2.5 Object Perspective

Object oriented (OO) modeling languages are generally built around objects, processes and classes, much like OO programming languages. Other concepts familiar to OO programmers like inheritance, instantiation, static/dynamic binding, broadcasting and polymorphism can also be found in modeling languages with this perspective. OO analysis can be quite comprehensive. It has been suggested that it should include class relationship models, class inheritance models, object interaction models, object state models and user access diagrams [85]. Many aspects from structural and behavioural modeling can be found in OO modeling; there are even aspects of static rule-oriented concepts. OMT was one of the earliest OO modeling approaches when it came out in 1991. The approach consisted of modeling languages for object models, dynamic models and functional models. Other impactful variants of OO modeling are Coad-Yourdon, Shlaer-Mellor and Fusion. UML is built on OO principles, but has been argued to be a multi-perspective language. We take a closer look at UML in the context of multi-perspective methods in section 2.3.2.3.

2.1.2.6 Communication

Most of the work within this perspective is based on language/action theory from philosophical linguistics and speech act theory. Speech act theory is often labelled as a 'meaning in use theory'. Early research on the topic was done by Searle (associates meaning with a limited set of rules for how an expression should be used to perform certain actions), and Wittgenstein (meaning is related to the whole context of use and not only a limited set of rules) [65]. The basic idea in the communication perspective is that people cooperate within work processes through their conversations and through mutual commitments taken within them. The illocutionary act consists of three parts; illocutionary context, illocutionary force and propositional context. The act consists of a speaker, hearer, time, location and circumstances. The speaker can generally raise three claims: truth (referring an object), justice (referring a social world of the participations) and claim to sincerity (referring the subjective world of the speaker). Examples of communication-based modeling approaches include Action Workflow [49], 'conversation for action' [87] and 'speech act models' (exemplified in [45]). KQML [23] and ACL [24] are two modeling languages that fit in this perspective.

2.1.2.7 Actor and Role

The perspective is utilised to provide descriptions of organisational and system structure. An actor is typically defined as a phenomenon that influences the history of another actor, whereas a role can be defined as the behaviour which is expected by an actor, amongst other actors, when filling the role. Modeling with these perspectives is based on organisational science, work on (object-oriented) programming languages (e.g. [75]), and work on intelligent agents in AI (e.g. [66]). Example languages are i^* [91] and e^3 value [29].

Example Language: i^*

i^* or i^* framework is a modeling language suitable for an early phase of system modeling in order to understand a problem domain. It was first proposed by Yu and Mylopoulos in 1994 [91]. The goal was to provide a more systematic approach to the design of business processes by providing appropriate representations of the knowledge that is needed for understanding and for reasoning about business processes. The framework uses goals, rules and methods to support the systematic analysis and design of business processes. The i^* framework consists of three integrated languages to be used for organisational modeling:

- The Actor Dependency modeling language (ADM)
- The Agents-Roles-Positions modeling language (hereafter shortened to ARP)
- The Issue-Argumentation modeling language (also known as GRL (Goal-Oriented Requirements Language))

In *ADM* each node represents a social actor/role. Links between them indicates that a social actor depends on the other to achieve a goal. Models are set up with *depender - dependum - dependee* pairs. It is distinguished between four types of dependencies: (i) goal dependency, (ii) task dependency, (iii) resource dependency, and (iv) soft-goal dependency [45]. An *i** *ADM* model example can be found in fig. 6.4. The model is part of the evaluation material of this project.

The *ARP* language consists of a set of nodes and links. (Social) actor refers to any unit to which intentional dependencies can be ascribed (the term 'social actor' is used to show that the actor is part of a complex network of agents, roles and positions). A role is an abstract characterisation of the behaviour of a social actor within some specialised context or domain. A position is an abstract place-holder that mediated between agents and roles.

GRL is designed to support goal-oriented modeling and reasoning about requirements, especially the non-functional requirements. It allows to express conflict between goals and helps to make decisions that resolve said conflicts.

2.1.2.8 Topological

Models using this perspective focus on the topological ordering between its different concepts. This conceptualisation has a background in cartography and the CSCW field, where differentiating between space and place is an important factor (see [21, 32]). 'Space' is geometrical arrangements that might structure, constrain and enable certain forms of movement and interaction (i.e. my student workspace at NTNU campus), while 'place' denotes the ways in which settings acquire recognisable and persistent social meaning in the course of interaction (i.e. a student workspace). In the examples given above, the place-concept is viewed as an instantiation of a space. There exists no fully agreed upon definition of the terms. Philosophers such as Casey (1993, 1998 [15]) and Tuan (1975, [76]) have taken part in the debate. Tuan claims that the four dimensions physical, personal, social and cultural characterise a place. Some modeling approaches that allow for spatial factors to be taken into account are extended UML activity diagrams ([27]), floor plans ([53]), and relative temporal nearness models ([53]).

2.1.2.9 Other Views on modeling Perspectives

A classic distinction of modeling perspectives contained the structural, functional and behavioural perspectives [56]. Yang [89] identified the following perspectives based on work by Falkenberg [22] and Wand and Weber [80]:

- **Data perspective:** This is parallel to the structural perspective discussed in section 2.1.2.3.
- **Process perspective:** This is parallel to the functional perspective discussed in section 2.1.2.2.

- **Event/behaviour perspective:** The conditions by which the processes are invoked or triggered. This is covered by the behavioural perspective discussed in section 2.1.2.1.
- **Role perspective:** The roles of various actors carrying out the processes of a system. This is covered by the actor and role perspective discussed in section 2.1.2.7.

Curtis identified the following perspectives in his 1992 work [17]:

- **Functional:** What elements are performed (functional perspective).
- **Behavioural:** When and how elements are performed (behavioural perspective).
- **Organisational:** Where and by whom elements are performed.
- **Informational:** Represent informational entities (structural perspective).

As revealed here, each distinction of modeling perspectives have a lot in common. The eight perspectives outlined by Krogstie [45], as detailed above, can be seen as a experimental continuation or modernisation of the distinctions of old. Krogstie based his distinction on the work of Yang [89] (mentioned above), Bubenko et al. [12], the NATURE project [40], and the widely recognised Zachman Framework [72].

2.2 Enterprise Modeling

This section will provide a brief overview of current approaches to enterprise modeling, to outline the landscape to which the goal is to find a good position for GEMAL.

2.2.1 Enterprise Architecture Frameworks

Enterprise architecture frameworks take a wider approach than a typical modeling language, typically by looking at the enterprise as a whole. These frameworks are often characterised as having a holistic approach i.e. [63]), and may thus be a step in the direction of GEMAL and molecular/holistic modeling languages. We detail two impactful ones here:

2.2.1.1 The Zachman Framework

Created by the Zachman Institute for Architecture [72, 92]. The framework is simply a logical structure for classifying and organising the descriptive representations of an enterprise. These structures can be of value to the management of the enterprise, as well as to the development of the enterprise's systems. The simplest form of the framework depicts the design artefacts that constitute the intersection between the roles in the design process (i.e. owner, designer, builder) along with the product abstractions (material, process, geometry). The framework encourages

designers to answer what, how, where, who, when and why questions to help describe the affected artefact. It aims to be simple, comprehensive, a language, a planning tool, a problem solving tool and to allow for users to freely decide which technical tools and methodologies to apply.

2.2.1.2 ArchiMate

ArchiMate is partly based on the IEEE 1471 standard, and was originally specified by Lankhorst in 2005 [46]. It started out as a partner and government founded project in the Netherlands, and is currently being maintained by The Open Group [1]. It is an open and independent modeling language for enterprise architecture that is supported by different tool vendors and consulting firms. ArchiMate makes it possible to describe, analyse and visualise the relationships among business domains. It is being applied to business processes, organisational structures, information flows, IT systems and technical infrastructure. The goal is to help stakeholders design, assess and communicate the consequences of decisions and changes within and between the mentioned business domains.

ArchiMate has a layered and service-oriented look on architectural models (See figure 2.3). Higher layers make use of services provided by the lower layers. The most important relation between layers is formed by use of relations. An ArchiMate model was used in this project's evaluation material (see fig. 6.5).

2.2.1.3 Other Frameworks

Some other impactful Enterprise Architecture Frameworks are GERAM (University of Brisbane [9]), ARIS (IDS Hamburg [63]), CIMOSA Framework (CIMOSA GmbH [93]), DoDAF Architecture Methodology (The FEAC Institute [55]) and TOGAF Architecture Methodology (The Open Group [4]).

⁴Figure source: "ArchiMate framework" by Marclankhorst licensed under CC BY-SA 3.0. A similar figure can be found in The Open Group's specification of ArchiMate [2].

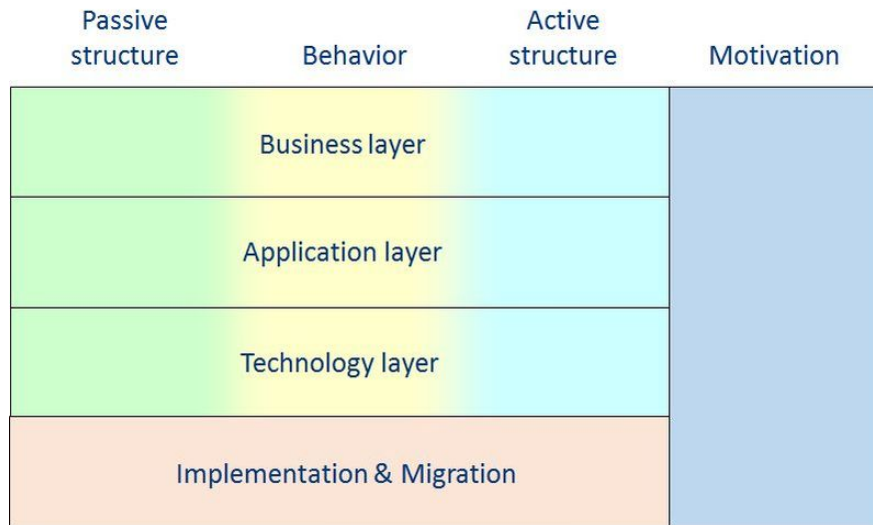


Figure 2.3: View model of the ArchiMate Architectural Framework, illustrating its layered and service-oriented approach to architectural models.⁴

2.3 Weaknesses of Current Approaches to Conceptual Modeling for Enterprises

H.D. Jørgensen stated a number of weakness found in most current and popular modeling languages as part of his 2004 PhD thesis [41]:

- Many languages are complex, containing numerous types and views not integrated in a systematic manner. This is especially the case for UML.
- In many cases mathematical, logical or technical concepts are applied instead of user or domain oriented. Petri nets and constraint based languages exemplify this.
- The languages that are precise and formal enough for automatic execution offer few opportunities for human contributions to interactive action. The languages do not handle process models with varying degrees of specificity.
- The semantics of language elements is generally static and not easily adopted to local contexts or multiple perspectives.

Jørgensen suggests that generic modeling techniques can help us meet these challenges in future approaches. Instance and property modeling can enable more flexible languages for local modification, multiple views and concept evaluation. He sees semantic holism as a promising, but poorly developed, framework for designing simple and flexible languages.

2.3.1 Holistic modeling

Most IS modeling languages are formal or semi-formal, and atomic semantics dominate. Typically the goal has been to identify model elements with a clearly defined interface that captures all the element's relevant relations to its environment. This *constructive* approach divides the problem into smaller sub-problems. When every sub-problem is solved, so is the problem as a whole.

In semantic holism, the meaning of each model element depends on the rest of the model. This is intended to make the language simpler, more flexible and user-oriented. The Routledge Encyclopedia of Philosophy defines semantic or mental holism as

"The doctrine that the identity of a belief content (or the meaning of a sentence that expresses it) is determined by its place in the web of beliefs or sentences comprising a whole theory or group of theories" [11].

Molecularism determines meaning and content in terms of small parts of this web (each element does not have to relate to *every* other element), while atomism defines elements independently of the rest of the web. This means that the conventional notion of semantic atomism claims that sentences have meaning independently of their relations to other elements, sentences, or beliefs. It has been argued that holism has a number of weaknesses [11]:

- It makes generalisation difficult, because sentences cannot be fully understood outside of their context.
- It makes it difficult for multiple theories to share any sentences or beliefs, because the meaning of those beliefs are contextualised in each theory.
- It makes logical reasoning, agreement, and translation difficult.
- It makes the semantics unstable, in that any change in one's attitude towards a sentence will change the meaning of the terms contained in it.

Supporters of the holistic approach have met these arguments by stating that a gradient of similarity of meaning can be allowed, replacing the dichotomy between agreement and disagreement. Two-factor theory is another approach, where the meaning of a sentence consists of an internal holistic factor, and an external referential factor that relates terms to real-world entities [11]. A moderate version of holism in which the mapping from beliefs to meaning of a sentence is many-to-one has also been introduced [39]. This implies that not every change in beliefs needs to affect the meaning of a sentence, even if it still may occur.

Semantic holism can be found in other areas than conceptual modeling. One example is the social construction of meaning that happen in communities of practice. The local meaning of terms are created inside the community, along with the role of boundary objects in cross-community relations. Boundary objects have a common identity across communities, but also

a specific local meaning inside each community (thus adhering to the two-factor theory) [41]. Semantic holism is used in biology, where ecosystems and genomes are described in holistic terms [30]. Functional grammar (FG) uses sentence holism consisting of predicates, predication, propositional content and clauses to construct sentences in which the meaning is expressed by a combination of all its elements [14]. There are also aspects of holism in traditional modeling languages, most visible through typical inheritance structures. Inheritance causes the meaning of one element to depend on the meaning of another (its superclasses).

H. Jørgensen claims that semantic holism is a promising strategy for human oriented modeling languages because reality as we observe it has clear holistic features, as observed in natural language and communication [41]. By allowing the meaning of different terms and clauses to be combined, one can articulate more nuanced knowledge than what is possible with atomic formalisms. By adding or removing from a sentence, one can better express flexibility, ambiguity and uncertainty. Jørgensen mentions the following as ways in which semantic holism can be used to simplify model articulation, in context of the web-based emergent workflow management system he worked on, WORKWARE [41]:

- Limited classification - one does not need specialised constructs for every concept as in atomic approaches. In this case holism can reduce complexity, improve flexibility, and ease reuse of model elements.
- Instances are referential terms - following the two-term theory of semantic holism, instances most often points directly to real world entities or phenomena that are of relevance to the model interpreter. This means that instance modeling facilitates human and social interpretation of the meaning of the model.
- Properties modularise aspects of meaning - dynamic addition of properties (or even 'things' in the context of GEMAL) allows users to articulate facts and meanings in a flexible manner. Properties can also be used to denote uncertainty, vagueness and incompleteness (example of this in the SeeMe notation, see section 2.3.2.4).
- Derived properties enable contextual semantics - properties may be calculated automatically based on the environment of a model object. Through this technique, a model object can in principle affect any other object in the model.
- Constellations - by grouping objects together in constellations, one can control how precise the composite object becomes. Individual objects in a constellation can be derived from the whole.

One of the arguments for adopting semantic holism over traditional constructive atomic approaches, is that it is impossible to properly predefine all connections between sub-systems. Holism is better equipped for modeling wicked, incompletely understood problems, that have yet to be tamed and solved [41]. They support the process of problem solving, and not just

documenting its outcome. Researchers have suggested replacing black-box, closed constructive structures with open, reflective implementations and semantic holism [20, 43].

2.3.2 Holistic modeling in Context

Holistic modeling is not an isolated idea, there are other ideas motivated by similar problems in traditional conceptual modeling. These approaches have many similarities. We cover some relevant ones here.

2.3.2.1 Facet modeling

Facet modeling is an approach intended to combat the weaknesses of traditional conceptual modeling approaches caused by their orientation towards certain aspects, i.e. activities, resources, objects, or actors. Facet modeling seeks to make it possible to freely represent aspects of a problem domain phenomena from a wide and extensible range of available aspects [58]. Different aspects can be co-represented whenever needed, and semantical relation between aspects are reflected in the created domain models. The modeler will have freedom from perspective, and can extend models with new kinds of aspects that are recognised as relevant to the problem at hand during analysis. Note that facet modeling is not built to prevent perspective and orientation in modeling. It is recognised that conceptual modeling should be based on a fundamental method or a set of guidelines, but these should be evolved through discoveries made in analysis instead of being implicit restrictions that come from the modeling language used.

Figure 2.4 shows a simple modeling concept (or pheonomena) called an item, modeled with facets. The 'cube' may be rotated to reveal other facets of the item. Each facet may represent one aspect or set of properties. Which facet that is most relevant at the time may depend on the context of the modeling session.

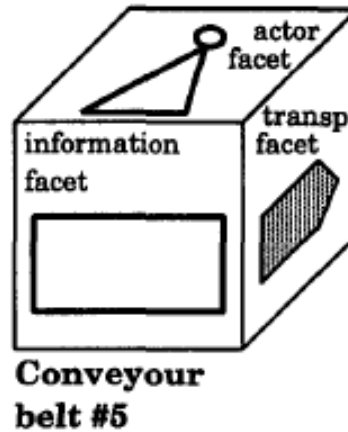


Figure 2.4: An example of an item with several facets, illustrated on each side of its cube. The cube can be rotated to reveal other facets. **Source:** [58]

2.3.2.2 Integrated use of Enterprise and IS Models

The Unified Enterprise Modeling Language (UEML) aims to support integrated use of enterprise and IS models expressed using different languages. The BWW model is taken as an outset. The goals of UEML are:

- **Exchanging information contained in enterprise and IS models across modeling language boundaries.** This is the central motivation behind UEML, which explains its focus on interoperability between modeling languages as a prerequisite for integrated use of the models that are expressed in those languages.
- **Creating new problem- and/or domain-specific methods** by combining elements from existing modeling techniques.
- **Systematic, quality-driven, reuse** of existing enterprise IS modeling languages.
- **Defining a core language for enterprise and IS modeling** (long-term goal; UEML may become a baseline for a new enterprise/IS modeling language once it stabilises).
- Facilitating a **web of languages and of models** (long-term goal).

UEML was first populated with a set of initial classes, properties, states, and transformations derived directly from the BWW model. Since then, it has grown and more constructs have been added. Currently, modeling languages including ARIS, BMM, BPMN, coloured Petri nets, GRL, IDEF3, ISO/DIS, and KAOS are incorporated in UEML [45].

2.3.2.3 Multi-Perspective Methodologies

Multi-perspective methodologies have existed for some time. They provide multiple partial modeling languages of which more than one is commonly to be used in the same IS development phases, for the same modeling worlds, and at similar modeling levels [58]. An older example can be found in the work of Olle et al. [56], while newer approaches include UML and EEML.

EEML is primarily geared towards generation of process support environments, but can also be used for more general enterprise modeling. EEML has the following sub-languages, with well-defined links between them: (1) process modeling, (2) data modeling, (3) resource modeling, (4) goal modeling [45]. An example EEML model can be seen in fig. 2.5.

UML is a language for specifying, visualising, constructing and documenting the artefacts of software systems, as well as for business modeling and modeling of other non-software systems [5]. Over the last 15 years, UML has taken a leading position in the area of object-oriented modeling in analysis and design. UML 2.4.1 has 14 diagram types, as illustrated in fig. 2.6. These are divided into two categories: (i) Structure Diagrams, which emphasise the things that must be present in the system being modeled (extensively used for documenting software architecture of software systems), (ii) Behaviour Diagrams, which emphasise what must happen in the system being modeled (extensively used to describe the functionality of software systems). Interaction Diagrams are a sub-set of Behaviour Diagrams, in which the flow of control and data among things in the system being modeled is emphasised.

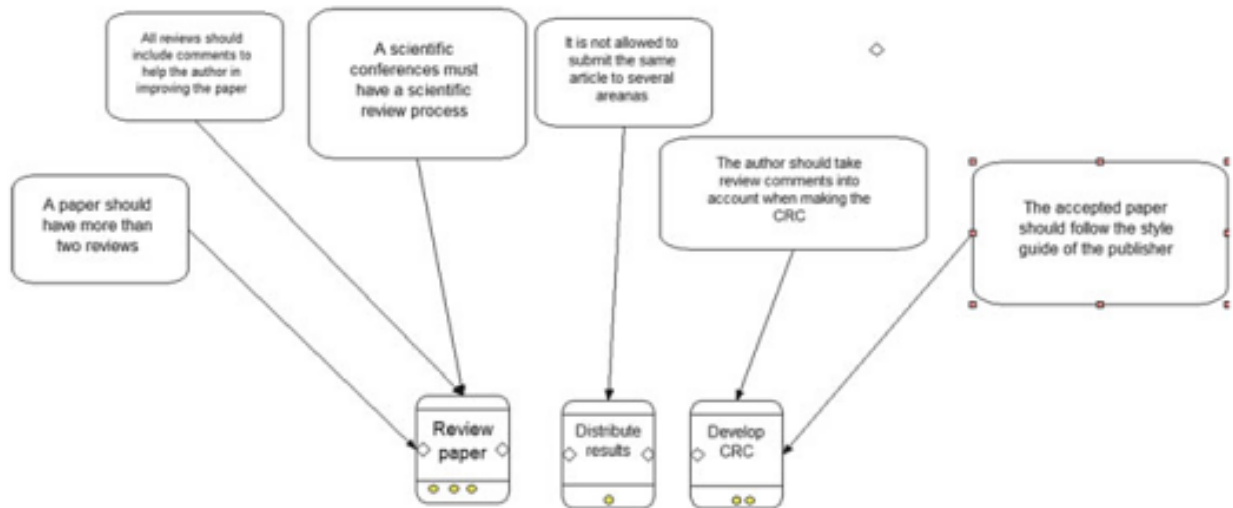


Figure 2.5: Relationship between goals and tasks in EEML **Source:** [45]

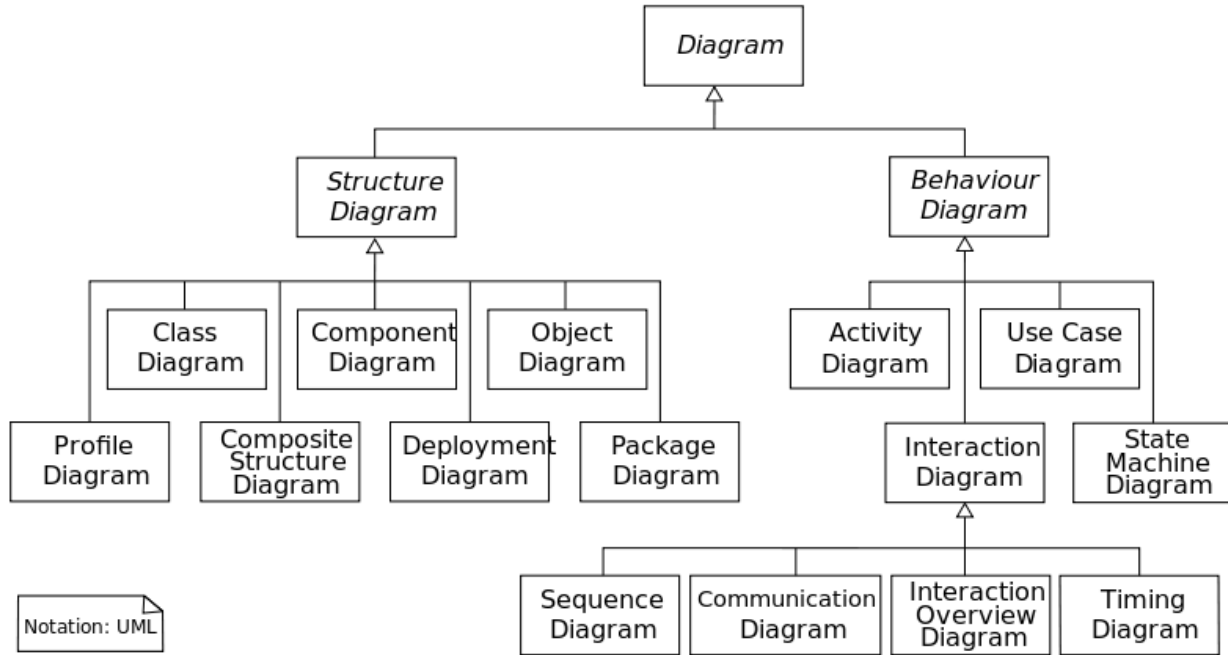
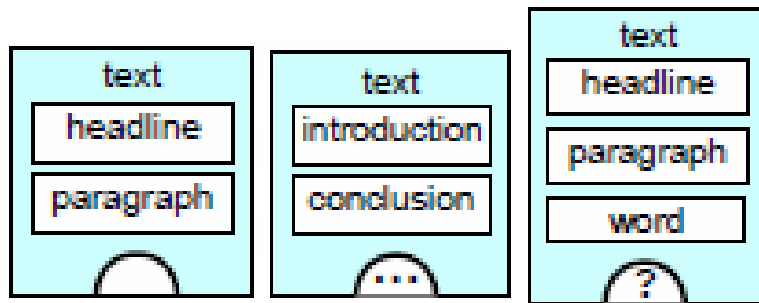


Figure 2.6: Overview of UML diagrams in UML 2.4.1

2.3.2.4 Example Notation: SeeMe



(a) Intentional incompleteness (b) More knowledge needed (c) Doubtful about correctness

Figure 2.7: Three types of 'mouseholes' used to model incompleteness in SeeMe. Source: [34].

SeeMe (semi-structured, socio-technical Modeling Method) is a modeling notation specialised on modeling socio-technical systems that was developed in 1997. It seeks to achieve a smooth communication process about socio-technical solutions, typically through a series of workshops or Socio-technical walktrough processes (STWT) [34]. The language is intended to be used in early stages of planning socio-technical systems, that is, systems containing complex interdependencies between people, humans and computers, and between technical components. It does not require the modelers to select a viewpoint, but allows the combination of view such

as processes, functions and tasks of objects. A SeeMe model can start as a sketch, where details are added in later stages as appropriate. The notation is built to support discussion and negotiation amongst stakeholders, it is not only there to document the results like many traditional modeling languages.

Something that separates SeeMe from traditional modeling languages is its dedicated constructs for modeling incompleteness and vagueness. The language has three basic elements; role, activity and entity. Compositions of elements are supported, as are relationships between elements and logical structures consisting of connectors, events and modifiers. Figure 2.7 depicts three ways to model uncertainty on entity elements. Similar functionality is available for roles, activities, and even relationships.

Holism can be observed in SeeMe in the way concepts are built into complex composite structures bound together by relationships and events, exemplified in [36]. SeeMe has been applied to a number of process modeling case studies [35, 36] and groupware training sessions [37], in which the researchers learned that incomplete models are simpler and easier to understand for the process participants. Also, SeeMe is holistic in approach because complex components can be decomposed freely.

2.4 Evaluation of Modeling Languages

Siau and Rossi reviewed various evaluation techniques for modeling methods used by both researchers and practitioners [68]. Their research lead to a method categorisation consisting of three classifications: feature comparison, theoretical and conceptual evaluation, and empirical evaluation.

Furthermore, analysis of the underlying philosophies and assumptions of these evaluation techniques lead to a proposed framework intended to identify the methods evaluation techniques by positioning them along two dimensions, ontology and epistemology. Ontology is concerned with the essence of things and the nature of the world. Some evaluation methods may belong on the realistic side of the scale ("the universe comprises objectively given, immutable objects and structures"), while others are nominalistic ("there is no given immutable 'out there', reality is socially constructed"). The other dimension, epistemology, is the grounds of knowledge relating to the way in which the world may be legitimately investigated and what may be considered as knowledge and progress. The two extremes of this dimension imply that the scientific method can be used to investigate the causal relationships (positivism), or that there is no single truth that can be proven by such investigation (interpretivism). No other broad attempts at reviews of existing method evaluation techniques that can compare to the one done by Siau and Rossi was uncovered in the literature survey of this project, but two conference series that are dedicated to the topic are mentioned in literature [68] (e.g. Method Engineering,

Workshop on Evaluation of Modeling Methods in Systems Analysis and Design, and CRIS)

The remainder of this section will further investigate the different evaluation techniques in use, while focusing on those that may be of particular interest to the conceptual modeling domain.

2.4.1 Feature Comparison

The studies in this category have typically been based on the idea of using different methods to model the same domain (e.g. the conference organisation case), and to determine how the various methods tackle the same problem. Early attempts were made to evaluate competing modeling languages by comparing their similarities and differences (e.g. Olle et al. in 1982, 1983, 1986 [81]), while others undertook case studies (e.g. Floyd in 1986 [25]). However, the absence of theory made it difficult to validate results. Feature comparison techniques have been criticised for their subjectivity. Both coming up with checklists to measure against and performing the evaluation are rather subjective tasks. On the upside, the evaluation can be relatively easy to perform if the included criteria are well-defined [68].

2.4.2 Theoretical and Conceptual Investigation

Siau and Rossi put evaluation techniques including metamodeling, metrics analysis, paradigmatic analysis, contingency identification, ontological evaluation and cognitive evaluation into this category [68].

2.4.2.1 Establishing a Theoretical Basis

As introduced in section 2.1.1, there was a lack of theory on which to validate research results of early conceptual models. Following that, the early 1990s spawned attempts at establishing a theoretical basis for conceptual modeling grammars. Much work has since followed the work of Wand and Weber [80], where they proposed that conceptual modeling grammars should be based on a theory of ontology. That is, a theory that articulates those constructs needed to describe the structure and behaviour of the world in general. Other similar work was done in the same period (see [86, 6]). This provided the opportunity to evaluate the ontological expressiveness of grammars, by comparing their constructs against the constructs in an ontology. Theoretical bases using other approaches than ontology have been developed, including concept theory[59], speech-act theory[7] and semiotics [44, 73].

2.4.2.2 Metamodeling and Semiotics

Model detonations are signs, and thus many in the field base work on modeling on theories from semiotics (e.g. [44, 73]). Semiotics, the study of signs, has been associated with philosophical and linguistic enquiry into language and communication from the time of the greek philosophers. Modern semiotics, as developed by Morris [52], describes the study of signs in terms of its logical components. The representation of a sign has an intended (referent) meaning and an interpretation (or received meaning). Morris further describes this in terms of syntactic, semantic, and pragmatic semiotic levels. Interpretation of a sign depends on sociolinguistic context (societal and linguistic norms) and the circumstances of affected individuals.

The SEQUAL framework, which has been used in the work of this thesis, is a systems modeling reference model for evaluating the quality of models. It has been developed by Krogstie and others since the 1990s [45]. SEQUAL looks at the quality of models relative to seven levels: physical, empirical, syntactic, semantic, pragmatic, social and deontic. By including aspects of affected models, body of knowledge, domain, modeling language, activities, actions and modeling itself, it seeks to guide a better understanding of models and modeling languages. This top-down quality framework is based on semiotic theory and the work of Morris [52].

Other frameworks in the same category includes Guidelines of Modeling (GoM) and Quality of Modeling (QoMo). GoM is focused on managing the subjectivism involved when building models. The framework consists of two dimensions; one for the range of model use (reference models for organisations and industries), and one for the degree of precision or concretion (general, system views, and description language) [64]. QoMo is focused on knowledge state transitions, cost management, and goal structure. The goal structure is directly linked to concepts of the SEQUAL framework [77].

Even though feature comparison and metamodeling has its similarities, it has been argued that the latter technique is more objective in nature. This is because the underlying work is based on modeled characteristics rather than on an ad hoc compilation of checklists and the identification of features based on the vague documentation on various modeling methods [74].

2.4.2.3 Metrics Analysis

Metrics analysis is aimed at analysing the complexity-based features of methods based on a standardised set of method metrics, as proposed by Rossi & Brinkkemper [62]. A formal meta-model of a method is analysed, and metric values are computed as a result. The values may then be compared to reference values provided in other research. Keng and Rossi claim that metrics can provide a valuable aid for comparing methods, but that more empirical work is needed to validate and fine-tune the metrics [68].

2.4.2.4 Paradigmatic Analysis

Paradigmatic analysis is about studying the assumptions behind systems development methods. The analysis form seeks to distance itself from the technical focus of other methods, attempting to widen the scope to fathom underlying assumptions of modeling methods on organisation development process and nature of the systems under development. An example framework for method analysis in this field can be found in Iivari and Kerola's work [38].

Wood-Harper & Fitzgerald published a taxonomy that classified methodologies in terms of what paradigm they used. Science paradigm and systems paradigm was identified as the two basic paradigms in use [88]. It has been claimed that paradigmatic analysis is less applicable to end users of modeling, but that it can be of aid when selecting a method for use within an organisation [8].

2.4.2.5 Contingency Identification

The contingency identification technique to methods selection uses heuristics to minimise risks and identify problems that are to be addressed by the methods. Davis proposed the following as criteria based on project contingencies for selecting amongst modeling methods: (1) The problem under investigation, (2) the people who perform the investigation [18]. It has been suggested to use contingency techniques along with other techniques, such as paradigmatic analysis and future analysis, into a so called 'multi-level' technique [8]. However, contingency identification techniques are only applicable when there are specific modeling tasks available to consider.

A way of doing heuristic evaluation is by conducting a so called *expert evaluation* (also called expert review [3]). In this method, an artifact is evaluated by experts working either separately or in teams. Experts often have access to relevant published research data, industry-accepted principles (heuristics) and best practices, and may have experience with observing users in lab and field settings. Expert evaluations in a professional setting are most common for usability testing of software systems, but examples of expert evaluations in the field of conceptual modeling can also be found (i.e. an evaluation of the benefits of using domain-specific modeling languages by Wegler et al. [84]).

2.4.2.6 Ontological Evaluation

As touched upon earlier in this section, Wand and Weber are reckoned as the proposers of ontological concepts to study systems analysis and design modeling methods [79, 80, 82]. The Wand and Weber model is based on and adapted from the work of Bunge [13], and is often referred to as the Bunge-Wand-Weber (BWW) model. The basic idea of ontological evaluation is to evaluate the constructs of a method by matching them with ontological constructs. Terms such

as construct overload, construct redundancy, construct excess and construct deficit have been introduced [80]. A benefit of the ontological approach is that it is derived from a philosophical foundation (Bunge's ontology). An example of use can be found in the work of Opdahl & Henderson-Sellers [57].

Note that GEMAL is based on the BWW model, which we will get back to in chapter 4. The SEQUAL framework uses BWW as a possible baseline for needed expressiveness in conceptual modeling [45]. The BWW terms mentioned above have been assigned to one of six categories of language quality (domain-, comprehensibility-, participant-, modeler-, tool- and organisational appropriateness). Construct incompleteness and construct excess are applied to the domain, and construct overload and construct redundancy are applied to comprehensibility appropriateness.

2.4.2.7 Cognitive Evaluation

Cognitive evaluation is concentrated around the importance of understanding the cognitive aspects of modeling. Keng Siau has been central to the development of this approach. He suggests using cognitive psychology as a reference discipline for information modeling and method engineering [67]. He also contributed to the GOMS (Goals, Operators, Methods and Selection Rules) technique, as exemplified in the evaluation of the nine diagramming techniques in UML by Siau and Tian [69]. Cognitive mapping techniques can be used to describe mental images that subjects use to encode knowledge and information. Cognitive evaluation techniques are well tested and documented in their respective disciplines, however, it can be a challenge to tailor them to fit the conceptual modeling field.

2.4.3 Empirical Evaluation Techniques

Empirical evaluation techniques use observation and propositions based on sense experience to evaluate systems and designs. Such research is conducted to test a hypothesis. The central theme in scientific method is that all evidence must be empirical, that is, based on evidence.

Empirical evaluation techniques include surveys, laboratory experiments, field experiments, case studies, action research, and verbal protocol [68]. We will not detail each technique here, because the techniques are less specific to the conceptual modeling domain than those in the forementioned categories. However, we will outline some perceived strengths and weaknesses connected to the survey technique, since it will be utilised in this project.

2.4.3.1 Surveys

Surveys are typically created with the purpose of gathering data on attitudes, opinions, impressions, and beliefs of human subjects via questionnaires. A priori hypotheses can be tested, and

an iterative approach can be followed to generate new hypotheses. A major challenge connected to using surveys is the low response rate, which typically ranges from a low of a few percentage points to a high of 20-30% [68]. An example of use of surveys to gather data on the topic of conceptual modeling can be found in the work of Gemino and Wand [26]. They used questionnaires to evaluate the effectiveness and efficiency of DFD and OO methods over a 3-year period; they solved the problem of low response rate by using students as respondents. Another weakness of the survey technique is that it generates perceptual rather than objective measures, but in some cases getting responses that say something about perceived usefulness and advantages can be of value. Surveys are not as controllable as some other empirical evaluation techniques, and are typically used to gather perception information from many geographically dispersed practitioners.

Chapter 3

Research Design and Methodology

This chapter describes the research model followed in this thesis, with its constructs and design guidelines. We start by introducing the research questions that have guided the work, before continuing to research approach and data validity assessment methods.

3.1 Conceptual Framework

3.1.1 Research Questions

1. **RQ1:** Is it possible to implement a molecular modeling language such as GEMAL in Metis?
2. **RQ2:** Is GEMAL able to represent every relevant aspect of traditional modeling cases?
3. **RQ3:** Are all notations when combining concepts freely like in GEMAL meaningful?
4. **RQ4:** Is the comprehensibility appropriateness of a molecular modeling language like GEMAL better than what can be achieved by combining traditional modeling languages?

The first research question deals with the implementational aspect of this work. The aim is to implement the GEMAL modeling language with all its constructs and functions in the Metis modeling environment. For the language to be of any use to modelers, a working implementation in a tool is required. An implementation is also useful when seeking to acquire empirical research data.

With the second research question, it is sought to explore the expressiveness of the GEMAL language. If the language is proven able to represent traditional modeling cases in an efficient way, one can begin to consider if its approach can contribute to developing the conceptual modeling scene.

The third research question is aimed at uncovering whether the idea of a modeling notation that puts few limits on the combination of items and properties can work in practice. It is sought

to find if restrictions are needed in order to encourage meaningful and understandable models and notations.

Finally, the fourth research question is aimed at uncovering whether the free modeling approach of GEMAL lends itself better to social actor interpretation than traditional modeling languages. Answers to this question will be sought through empirical evaluation techniques and a theoretical and conceptual investigation of available material.

3.2 General Research Approach

3.2.1 The Design Science Research Process

The design science paradigm seeks to extend the boundaries of human and organisational capabilities by creating new and innovative artifacts. Knowledge and understanding of a problem domain and its solution are achieved in the building and application of the designed artifact. The rigorous design science research process involves designing an artifact to solve a problem, evaluating the design, making research contributions, and communicating the results to the appropriate audiences [78].

Figure 3.1 shows the design science research process, as illustrated in the work of Peffers et al. [60, 61]. The process is structured in sequential order, but it is possible to start at any of the four first steps. After evaluating the artifact in step five, researchers can choose to iterate back to step two if rethinking of the solution is needed, or step three if further improvements to the effectiveness of the artifact is needed. Another option is to continue to communication, and leave further improvements to subsequent projects.

The research model of this project followed a design and development centered approach. The main artifact (the GEMAL modeling language) had already been defined and made relevant as a proposal in another project, and as such the task was to develop and design it further, preparing it for demonstration and evaluation. However, the artifact had not been previously evaluated, so an initial evaluation was conducted with the goal of assessing the quality of previous work and obtaining a better understanding of how it can be improved upon.

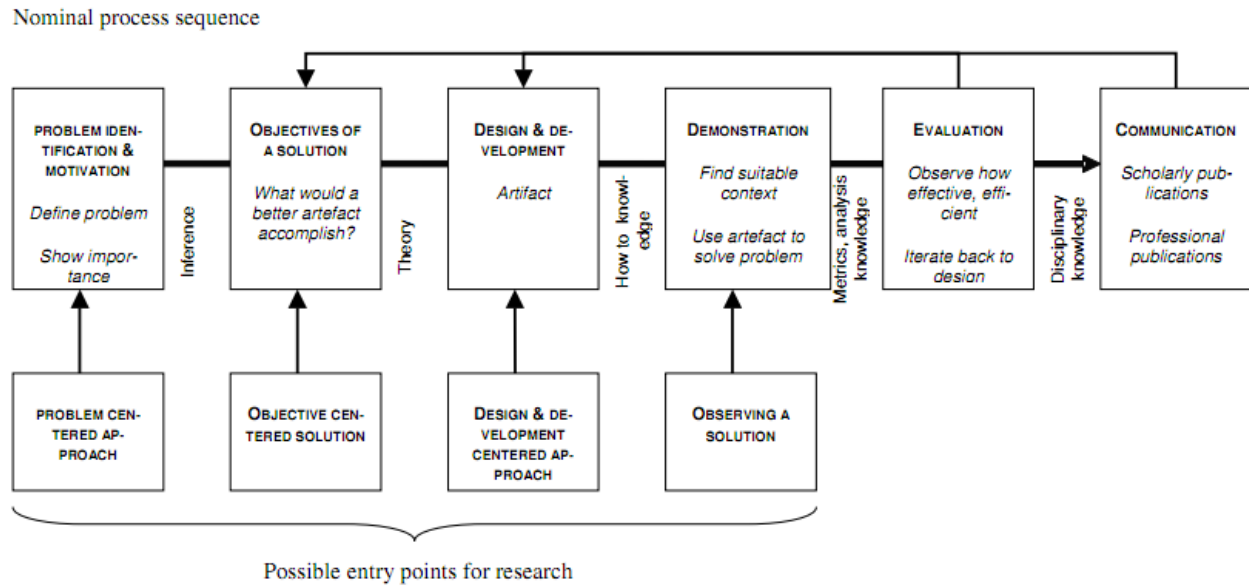


Figure 3.1: The design science research process [60].

Guideline	Description
Guideline 1: Design as an Artifact	Design-science research must produce a viable artifact in the form of a construct, a model, a method, or an instantiation.
Guideline 2: Problem Relevance	The objective of design-science research is to develop technology-based solutions to important and relevant business problems.
Guideline 3: Design Evaluation	The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.
Guideline 4: Research Contribution	Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies.
Guideline 5: Research Rigor	Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.
Guideline 6: Design as a Search	The search for an effective artifact requires utilising available means to reach desired ends while satisfying laws in the problem environment.
Guideline 7: Communication of Research	Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.

Table 3.1: Design science guidelines [78].

To ensure that the research was conducted in a sound manner relative to design science, it was strived to follow the seven design science guidelines developed by Hevner et al. [78], as shown in table 3.1:

- The first guideline states the need of producing a viable artifact. In this project, the GEMAL modeling language is the main artifact.
- The second guideline is about problem relevance. The relevance of this project is explained in the motivation part of the introduction (section 1.1), and further explored in the discussion of section 2.3.
- The third guideline fathoms evaluation. The main artifact went through both an early theoretical evaluation, and an empirical evaluation after its technical implementation. In addition, it was evaluated relative to case studies.
- Relative to the fourth guideline, the main research contribution of this project is the implementation and evaluation of the GEMAL modeling language.
- This chapter explains the methods used to develop and evaluate the prototype, research model and guide the implementation, and ensure that it was done in a rigorous way (guideline five).
- Guideline six states that research should be carried out as a search, which has been done from the start in this project. It started out by finding means to evaluate the language proposal that was available, before mapping best practices and alternatives available for implementation and evaluation of modeling languages.
- Relative to guideline seven, the results of this project have been communicated through this report.

3.2.2 Designing for modeling Language Quality

3.2.2.1 The SEQUAL Framework

To ensure adequate quality when evaluating GEMAL, SEQUAL has been used as a reference model. The SEQUAL framework ('semiotic quality framework') is a systems modeling reference model for evaluating the quality of models [45]. The framework follows a top-down quality approach. We discussed SEQUAL in relation to other techniques for evaluation of modeling languages in section 2.4. Here, we outline the topics covered by SEQUAL [45] (refer to fig. 3.2 for a graphical overview):

- **Model activation** - the process by which a model affects reality. It involves actors interpreting the model and to some extent adjusting their behaviour accordingly. The process can be automated, manual, and interactive.

- **Sets in the quality framework** - A (actors), L (what can be expressed in the modeling language), M (what is expressed in the model), D (what can be expressed about the domain), K (the explicit knowledge of the participating persons), I (what the persons in the audience interpret the model to say), T (what relevant tools interpret the model to say), G (the goals of the modeling).
- **Physical quality** - Main aspects of physical quality include (i) externalisation (is it possible to externalise knowledge by using the model language?), (ii) internalisability (about model persistence), and (iii) basically (is the model language able to express the model domain?).
- **Empirical quality** - The model is externalised in order for this to be evaluated. Main aspects of the topic include (i) Ergonomics, (ii) Readability, (iii) Layout, and (iv) Information theory. This topic seeks to find out if the model is in fact readable.
- **Syntactical quality** - Corresponds the model M and the language extension L of the language in which the model is written. Main aspects include: (i) Error Detection (check syntactical completeness upon user request), (ii) Error Prevention (making only constructs of the language's allowed vocabulary available), and (iii) Error Correction (replace a detected error with a correct statement).
- **Semantic quality** - What is expressed in the model? Semantic goals of the framework are validity (all statements in the model are correct and related to the problem. $M \setminus D = \emptyset$) and completeness (if the model contains all relevant and correct statements to solve the problem. $D \setminus M = \emptyset$).
- **Perceived semantic quality** - The relation between an actor's interpretation of a model and his/her knowledge of the domain. Goals are to reach perceived validity ($I \setminus K = \emptyset$) and perceived completeness ($K \setminus I = \emptyset$).
- **Pragmatic quality** - The correspondence between the model and people's interpretation of it. Comprehension is the only pragmatic goal in the framework (people that read the model need to understand it). Pragmatic quality relates to the effect the modal has on the participants and the world. Four aspects that are important to this topic include: (i) the human interpretation of the model (should be correct relative to what is meant), (ii) the tool interpretation of the model (should be correct relative to what is meant to be expressed), (iii) the participants learn based on the model, and (iv) the domain is changed (preferably in a positive direction relative to the goal of modeling).
- **Social quality** - The goal is to reach agreement about knowledge, interpretation, and model amongst participants. This is achieved if perceived semantic quality and comprehension are achieved. SEQUAL distinguishes between two agreement types for the concepts mentioned: relative agreement (all knowledge, interpretation, and models are consistent) and absolute agreement (all knowledge, interpretation, and models are equal).

- **Knowledge quality** - Encompasses the degree of internalisation of existing organisational reality. Knowledge in a domain can be complete ($D \setminus K = \emptyset$) or valid ($K \setminus D = \emptyset$). Knowledge quality can be improved by: (i) stakeholder identification, (ii) knowledge source identification, (iii) research and investigation, (iv) participant selection, (v) participant training, and (vi) problem definition.
- **Language quality** - To achieve good language quality it is important that: (i) the language is appropriate to the domain, (ii) the language is appropriate to the participants' knowledge of modeling languages, and (iii) the language is appropriate to express the knowledge of participants. Improving language quality can lead to improved interpretation for participants and technical actors.
- **Organisational quality** - Can be achieved by ensuring that all statements in the model contribute to fulfilling the goals of modeling (organisational goal validity), and that all the goals of modeling are addressed through the model (organisational goal completeness).

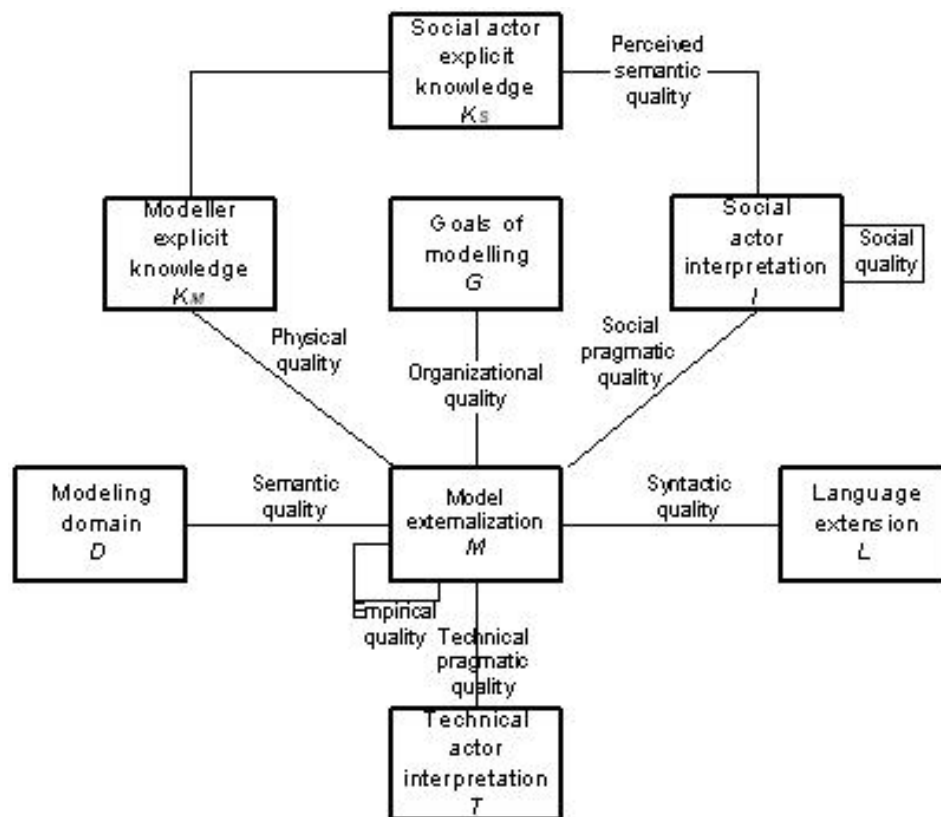


Figure 3.2: Illustration of the SEQUAL framework Source: [45]

3.2.2.2 Moody's Notation: a Scientific Basis for Constructing Visual Notations

Daniel L. Moody has been developing a scientific basis for constructing visual notations in software engineering, as shown in his 2009 article [51]. His focus is on the 'physics' of notations, and he created a set of principles for designing cognitively effective visual notations: ones that are optimised for human communication and problem solving. The principles were synthesised from theory and empirical evidence from a range of fields, and are grounded on explicit theories of how visual notations communicate. They can be used to evaluate, compare and improve existing visual notations, as well as to construct new ones. As a part of the work of this thesis, both cases will apply. Moody's principles will be used both to evaluate and improve the visual notation of GEMAL based on its language proposal, and be used to add new content to the language. Here we outline Moody's principles, refer to his article for details [51]:

1. **Semiotic Clarity (SC):** The principle of semiotic clarity states that there should be a 1:1 correspondence between semantic constructs and graphical symbols. When SC is broken, one or more of these four anomalies can occur: (i) *Symbol redundancy* (multiple graphical symbols can be used to represent the same semantic construct), (ii) *Symbol overload* (two different constructs can be represented by the same graphical symbol), (iii) *Symbol excess* (a graphical symbol does not correspond to any semantic construct), (iv) *Symbol deficit* (there are semantic constructs that are not represented by any graphical symbol).
2. **Perceptual Discriminability (PD):** Perceptual discriminability is the ease and accuracy with which graphical symbols can be differentiated from each other. Accurate discrimination between symbols is a prerequisite for accurate interpretation of diagrams. Moody mentions five factors that can impact human visual information processing (perceptual discrimination): (i) *Visual distance* (the greater the visual distance between symbols, the faster and more accurately they will be recognised), (ii) *The primacy of shape* (shape should be used as the primary visual variable for distinguishing between different semantic constructs), (iii) *Redundant coding* (similar to dual coding, using multiple visual variables to distinguish between symbols can increase the visual distance between them), (iv) *Perceptual popout* (visual elements with unique values for at least one visual variable can be detected preattentively and in parallel across the visual field), and (v) *Textual differentiation* (relying on text to distinguish between symbols).
3. **Semantic Transparency (ST):** With this principle, Moody suggests to strive for visual representations whose appearance suggests their meaning. Symbols can be *semantically immediate* (a novice reader would be able to infer its meaning from its appearance alone), *semantically opaque* (there is a purely arbitrary relationship between its appearance and its meaning) or *semantically perverse* (a novice reader would be likely to infer a different or even opposite meaning from its appearance). Symbols can be rated somewhere between immediacy and opacity as well.

4. **Complexity Management (CM):** CM refers to the ability of a visual notation to represent information without overloading the human mind. Complexity has a major effect on cognitive effectiveness, which is affected by human perceptual limits and cognitive limits. Moody suggests to combat complexity with abstraction mechanisms including *modularisation* and *hierarchies*.
5. **Cognitive Integration (CI):** Moody suggests that diagrams should include explicit mechanisms to support integration of information from different diagrams. He divides the topic into *conceptual integration* (mechanisms to help the reader assemble information from separate diagrams into a coherent mental representation of the system) and *perceptual integration* (perceptual cues to simplify navigation and transitions between diagrams).
6. **Visual Expressiveness (VE):** One should strive to use the full range and capacities of visual variables. Visual variation across the entire visual vocabulary is measured. This is different from PD (see above), that measures only pairwise visual variation between symbols. Moody mentions color as a cognitively effective visual variable that should be used more. Choice of variables should be tailored for the nature of the information to be conveyed, and graphical encoding should be preferred to textual encoding.
7. **Dual Coding (DC):** With this principle Moody suggests to use a combination of visual variables (in many cases one graphical and text leading to a *hybrid symbol*) to strengthen PD and VE.
8. **Graphic Economy (GE):** This principle suggests to keep the number of different graphical symbols cognitively manageable. The human mind can typically discriminate between perceptually distinct alternatives in around six categories, putting an upper limit for graphical complexity. Strategies for a manageable GE includes *reducing semantic complexity* (reduce the complexity of symbols in the language), *introduce symbol deficit* (choosing not to show some constructs graphically), and *increasing visual expressiveness* (adding more visual variables to increase human discrimination ability).
9. **Cognitive Fit (CF):** Use different visual dialects for different tasks and audiences. *Expert-novice differences* (problem solving skills) should be taken into account along with the *representational medium* the notation seeks to support (hand drawings on a whiteboard may impose extra considerations to PD, ST, VE).

Note that the perfect notation that scores well for every one of Moody's principles likely can't be made. Trade-offs have to be made, as scoring well according to one principle may decrease the scoring for another. Synergies can be exploited in cases where principles support each other.

3.3 Data Collection

An online survey and a set of expert evaluations were conducted to collect data in this research. Together, they cover two of the basic methods of data collection:

1. Personal interviews
2. Self-enumeration

Self-enumeration was used to collect data for the online survey. The respondent completes the survey without the assistance of the researcher. This method is less time consuming than personal interviews, so larger samples can be selected. Anonymity is guaranteed for the respondents. However, the respondent does not have anyone present that can clarify questions, so it becomes important to state questions in a clear and unambiguous way. Respondents are required to have a minimum level of computer literacy to be able to complete the survey on their own. There is typically a low rate of researcher bias in self-enumeration techniques. However, since the researcher decides who to distribute the survey to, he can have an impact on characteristics of the pool of respondents (i.e. age discrepancy, ethnicity, gender, culture). No effort was put in to attract respondents with any particular set of skills or experience level, and no specific age group was targeted, but the selection of medium for distributing the survey may have had an impact on the collected statistics.

Personal interviews were done for the expert evaluation. This form of evaluation is more time consuming than self-enumeration, and interviewer bias can have a great impact on the results. A strength with this method is that questions can be clarified if any misunderstandings are uncovered. Also, the researcher present at the evaluation can attempt to guide the evaluation towards topics that are of particular interest.

Chapter 4

Initial Overview of GEMAL

This chapter presents GEMAL (General Enterprise Modeling and Activation Language), the modeling language that was developed as a main part of this project. We start by having a closer look at the material published about GEMAL so far, before we analyse what form further work on the language should take.

4.1 Language Proposal as a Starting Point

GEMAL was first proposed as an enterprise modeling multi-perspective language by John Krogstie in 2012 [45]. A brief, informal overview of core concepts of the language was provided. GEMAL was stated to follow a molecular approach, as opposed to atomic or fully holistic approaches. The approach of the language was suggested as a way to combat issues stemming from the complexity introduced by current modeling approaches, exemplified by the daunting number of concepts of UML and BPMN, as well as the perspective combination approach used by UML and EEML.

4.1.1 Basic Concepts

Inspired by the ontology of Wand and Weber [79], 'thing' is the elementary concept of GEMAL [45]. The perception that the physical and social world is made up of things and specialisations of things is followed. Two or more things can be associated into a composite thing. Systems of things can be built, where couplings among subsets of things exist. Things may possess properties, belong to classes of things, and have a state and history. Things can exist both on a type level and an instance level. One can even create things that do not adhere to any specific type. In addition, things can be connected through different relationships.

The language proposal lists the following as properties that every thing will have: ID, Name, Description, Start-time, End-time, Instantiation level (type or instance), concept specialisa-

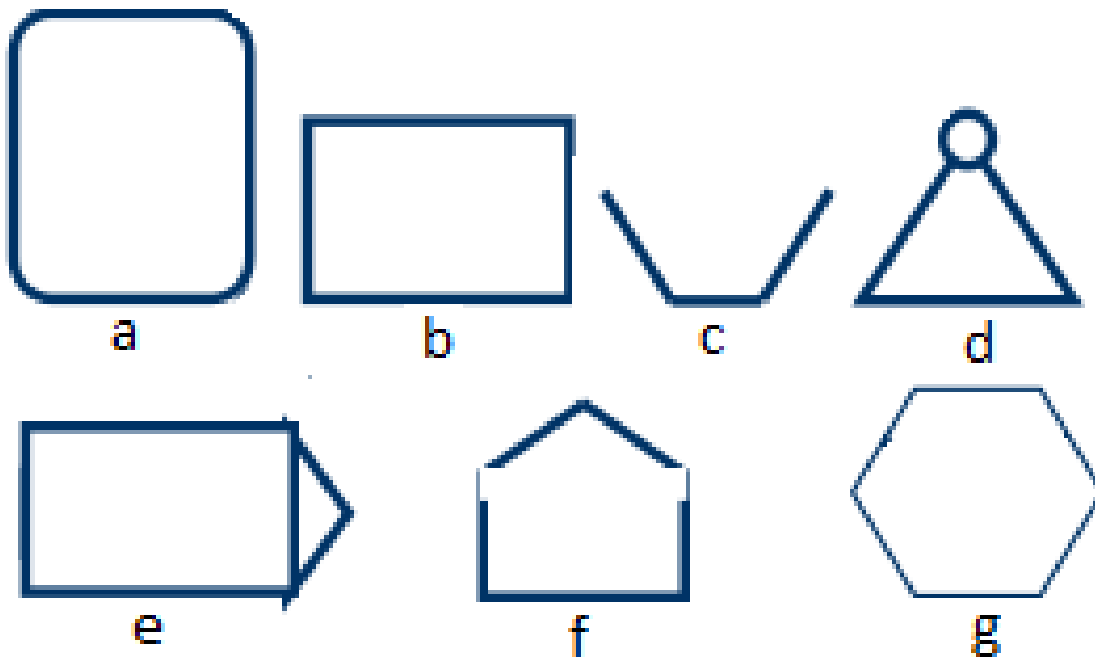


Figure 4.1: Symbols used for different specialisations of thing in the GEMAL language proposal. (a) Process, (b) product, (c) location (d) person, (e) goal, (f) organisation, (g) capability. **Source:** [45]

tions, modality (necessity, obligation, recommendation, permission, discouragement, prohibition, contradiction) and state (no state space defined).

Figure 4.1 shows the symbols proposed for different specialisations of the thing concept. The language proposal further details what each symbol means and how they are to be used [45]. Fig. 4.2 shows how composite objects can be visualised. Fig. 4.2a illustrates a process specialisation of a thing. The bottom part of every thing, separated by a horizontal line, is an area dedicated to visualising its sub-concepts. The top part is used for aggregation of the thing, here illustrated by sub-processes within the parent process. The circle labeled 'state' is used to visualise the state of the thing, and the very top identifies the thing with an ID or name. Fig. 4.2b illustrates a composite product in a similar fashion. The grey background means that the product is modeled at instance level.

4.1.2 Language mechanisms

Composites of things function as the language's main abstraction mechanism. Things can be closed like a typical container, hiding its internal details. This may be practical when seeking to get an overview of a situation at a higher level, where details may be of less interest. This is exemplified in fig. 4.3. Note that details of i.e. the 'proposal' object as seen in fig. 4.2b can be

viewed if needed. Also, sub-concepts can be further decomposed and specified, depending on the level of detail one seeks to include in a given model.

Relationship types mentioned in the language proposal include precede, support, govern and communicate. General (unspecified) relationships are also allowed. Relationships can be annotated with things or relationships to indicate its context. Relationships can further be used to build up generic logical relationships. Gateways (of types OR, AND, XOR) as seen in several enterprise modeling languages, are intended as a part of the language.

The meaning of concepts (thing and its specialisations) in GEMAL will change depending on the context in which they are used. For example, a process can be a production or usage process in the context of a product, the process that a person performs in the context of a person, a goal governance process in the context of a goal, etc. These context dependent interactions are detailed and further exemplified in the language proposal [45].

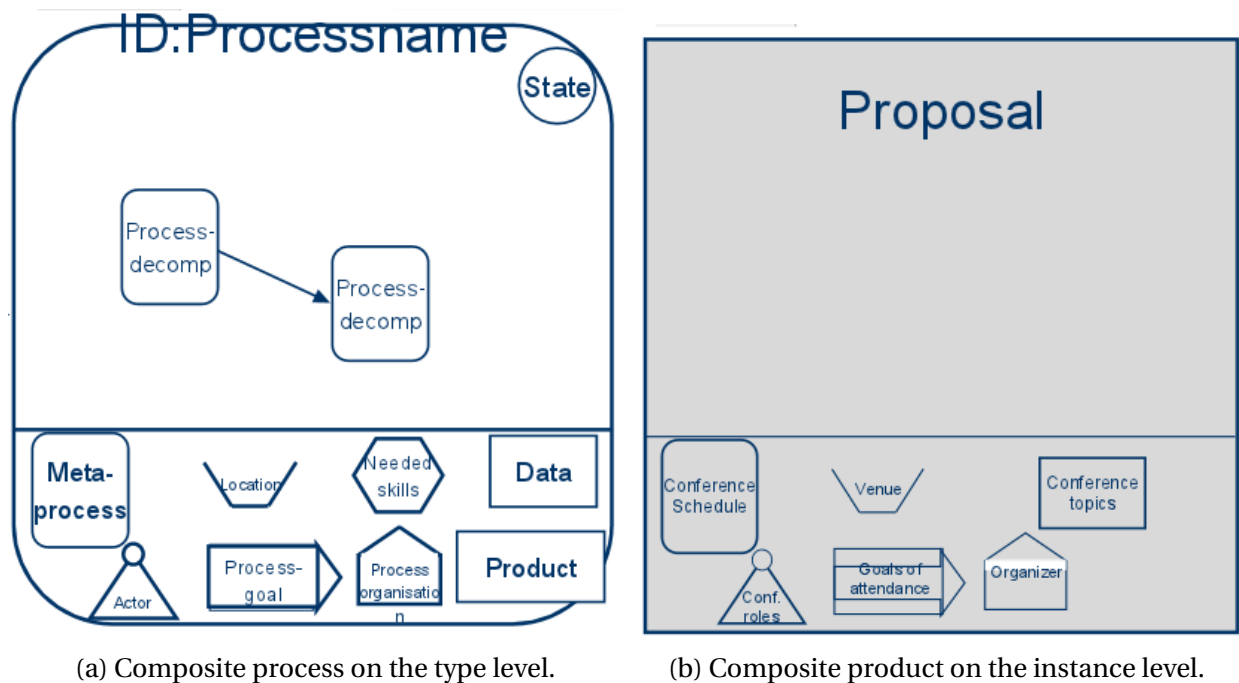


Figure 4.2: Examples of composite concepts, as illustrated in the GEMAL language proposal. Source: [45]

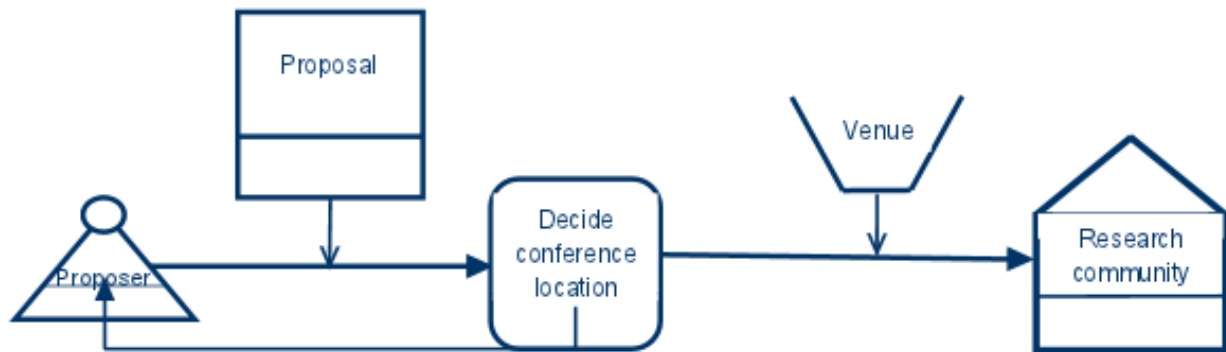


Figure 4.3: An example of a high level process in GEMAL, provided with the language proposal.
Source: [45]

4.2 Initial Evaluation of GEMAL's Language Quality

Before starting the implementation of GEMAL in the Metis modeling environment, an initial evaluation of the quality of the modeling language was done, as described in this section. The intention was to get a better understanding of its focus, strength and weaknesses, while also providing a chance to make improvements that it could be harder to make after the technical foundation has been made. Because of the limited content available for evaluation, empirical evaluation techniques were not used. Neither were feature comparison techniques, since at this stage it was more interesting to assess the quality of the language on its own merit, rather than comparing its quality to that of other languages. As such, the evaluation followed a theoretical and conceptual investigation outlook.

It was chosen to follow the SEQUAL framework as a general guideline. However, some aspects of the framework are unfit to be included in a high level language evaluation, and have been omitted. Modeler-, tool- and organisational appropriateness have not been considered at this stage, given the limited material available. In generic evaluations of modeling languages (i.e. where you do not look at some particular people using a modeling language for a particular task), the focus is typically on domain appropriateness, tool appropriateness (when automatic tool support is of importance) and comprehensibility appropriateness [28].

4.2.1 Domain Appropriateness

Domain appropriateness relates the modeling language and the domain. Ideally, the conceptual basis of a modeling language must be powerful enough to express anything in the domain, not having what Wand and Weber terms construct incompleteness [79]. The language should also avoid construct excess, that is, being able to express things that are not in the domain [79].

SEQUAL suggests two possible ways to evaluate domain appropriateness [45]; (i) look at how the modeling perspectives found useful for the relevant modeling tasks are covered, (ii) base the evaluation on an ontological theory.

GEMAL is somewhat of a special case when it comes to domain appropriateness, in that it seeks to avoid enforcing any modeling perspective. This makes it harder to compare GEMAL to the typical expressiveness of models of any given perspective. The following evaluation employs the second alternative proposed by SEQUAL, both because there are no concrete modeling tasks available that will fathom everything GEMAL can be required to represent, and because it seems very relevant to understand how GEMAL compares to the BWW conceptual model (see section 2.4.2.6). That is, since GEMAL was stated to be based on the BWW model in its language proposal [45]. The collection of BWW concepts in table 4.1 have been taken from Opdahl and Henderson-Sellers ontological evaluation of the UML using the BWW model [57], which they in turn had compiled from a series of previous work on the BWW model.

Table 4.1: Representational mapping of GEMAL constructs to BWW concepts

BWW concept	GEMAL representation
BWW-thing	GEMAL-thing
BWW-property [of a thing]	GEMAL-property. GEMAL properties are basically intrinsic properties [of a thing] because the properties are inherently a property of an individual thing. However, properties of composite things can be given meaning to by the context of the thing, making it closer to a BWW-mutual property.
BWW-property function [of a thing]	All the GEMAL properties that represent BWW-properties are also BWW-property functions. However, most of them are trivial, because they represent BWW-properties that do not change over time.
BWW-codomain [of a property function]	GEMAL-datatype. GEMAL values represents an element in a BWW-domain (i.e. necessity in the modality domain.).
BWW-intrinsic property [of a thing]	GEMAL-property represents a subtype of BWW-intrinsic property [of a thing].
BWW-mutual property [of two or more things]	GEMAL relationships represent a BWW-mutual property of two or more things. Intrinsic properties of composite things can be given meaning to by its context, and it may thus act as a mutual property [of two or more things].

Continued on next page

Table 4.1 – continued from previous page

BWW concepts	GEMAL representation
BWW-complex property [of a thing]	GEMAL-property with a non-primitive type represents a BWW-intrinsic complex property. 'Name' and 'State' properties of things are complex types, as they change the visualisation of the thing (by displaying on the thing).
BWW-law property [of a thing]	GEMAL-goal. Also, the modality property of GEMAL-thing and GEMAL-relationship are intended to be able to restrict the values of other properties. No further details on this in the language proposal.
BWW-natural law property	Alethic GEMAL-goal (Goal with the 'necessity' modality value).
BWW-human law property	Deontic GEMAL-goal (Goal with one of the available deontic modality values (obligation, recommendation, permission, discouragement, prohibition)).
BWW-class [of things]	(See BWW-natural kind.)
BWW-natural kind [of things]	Each specialisation of thing (process, product person etc.). Every specialisation of thing may act as a container for composite things, or be part of such a container. Every specialisation of thing may contain or be part of aggregation structures.
BWW-characteristic property [of a class or natural kind], BWW-property in general	No specialisation of GEMAL-thing has any specialised property type that applies to that specialisation alone. GEMAL-communication, support, and govern relationships represent mutual (binding) properties.
BWW-subclass [of things]	(See subkind.)
Subkind (sub-natural kind) [of things]	Specialisations of GEMAL-thing.
Natural kind/sub-kind relationship	GEMAL-generalisation structure.
BWW-state [of a thing]	GEMAL-state. The state property of [specialisations of] GEMAL-thing represents the state of a thing when it is acted on by one or more other things.
BWW-history [of a thing]	GEMAL does not visually represent the history of things, but it is stated in the language proposal that a things history is the trajectory of states of the thing.

Continued on next page

Table 4.1 – continued from previous page

BWW concepts	GEMAL representation
BWW-event [in a thing]	The GEMAL-precede relation can cover some types of events.
BWW-process [in a thing or system thing]	Every specialisation of thing may possess a process composite thing that describes its creation process, production, usage, development, maintenance etc.
BWW-transformation	GEMAL-process
BWW-state law [of a thing]	GEMAL-goal applied to a thing can impact the state law of things. GEMAL-govern relation allows one thing (or composite-thing) to restrict the state of another thing. The modality of a composite-thing or relationship may restrict the lawful states of other things.
BWW-transformation law [of a thing]	GEMAL-goal applied to a GEMAL-process.
BWW-external event [in a thing, subsystem or system]	GEMAL-precede relation can symbolise external events for a thing, subsystem or system.
BWW-internal event [in a thing, subsystem or system]	Not addressed in the GEMAL language proposal. Can likely be handled by internal processes in things.
BWW-stable state [of a thing]	GEMAL-thing's completed or canceled states.
BWW-unstable state	GEMAL-thing's not-started, on schedule, behind schedule states.
BWW-conceivable state space [of a thing]	Can be shown by surrounding composite things for a GEMAL-thing.
BWW-possible state space [of a thing]	GEMAL has a limited number of allowed states for things.
BWW-lawful state space [of a thing]	No specific counterpart in GEMAL. Will possibly be implied by the state of other things.
BWW-conceivable event space [of a thing]	No specific counterpart in GEMAL.
BWW-lawful event space [of a thing]	No specific counterpart in GEMAL.
BWW-coupling [of things], BWW-acting on [another thing]	GEMAL composite structures and relationship couplings.
BWW-binding mutual property, BWW-direct acting on	Annotating things and relationships with communication and support relationships.

Continued on next page

Table 4.1 – continued from previous page

BWW concepts	GEMAL representation
BWW-coupled event	No specific counterpart in GEMAL.
BWW-composite thing	GEMAL composites of things built into systems or sub-systems. GEMAL aggregates of things.
BWW-component thing	Every GEMAL thing can be a composite thing.
BWW whole-part relation [between things]	GEMAL-aggregation (when the whole is a non-system thing). GEMAL-composition.
BWW-resultant property [of a composite thing]	No specific counterpart in GEMAL.
BWW-emergent property [of a composite thing]	No specific counterpart in GEMAL.
BWW-system [of things]	GEMAL-composition allows building systems of things.
BWW-system composition	Every GEMAL-thing can be parts of a system.
BWW-system environment	Communicate, govern, support relationships between things in a system and things outside it.
BWW-system structure	No specific counterpart in GEMAL.
BWW-subsystem	GEMAL-composite things in a system may have its composition and structure as part of another system.
BWW-system decomposition	GEMAL-systems may be built from a series of sub-systems.
BWW-level structure	GEMAL nested decomposition reveals the partial order over the subsystems that are part of a decomposition.

This evaluation could have continued with a inverse interpretation mapping from GEMAL concepts to BWW representation to add more detail, as was done in Opdahl and Henderson-Sellers evaluation of the UML [57], but it seems that table 4.1 has summarised most representation found in the GEMAL language proposal.

Wand and Webers concept of semiotic clarity (SC) states that there should be a 1:1 correspondance between semantic constructs and graphical symbols. This may be undermined by four ontological discrepancies [80]. which we will consider here:

1. **Construct overload** - When a modeling construct corresponds to several ontological concepts. The analysis in table 4.1 did not uncover any serious violations of this type.
2. **Construct redundancy** - When several (overlapping) modeling constructs represent the same ontological concept. No cases of this was found in the overmentioned analysis.

3. **Construct excess** - When a modeling construct does not represent any ontological concept. No cases of this was found in the analysis (could have been made more evident if a inverse interpretation mapping was conducted).
4. **Construct deficit** - When an ontological concept is not represented by any modeling construct. Several examples of this was found in table 4.1 on topics including event and properties. GEMAL may struggle with representing models requiring representations of these kinds.

4.2.2 Comprehensability Appropriateness

Comprehensability appropriateness encompasses the social actor interpretation of the language. The goal is that participants of a modeling environment using the language understands all the possible statements of the language. To assess comprehensability appropriateness at the conceptual level, SEQUAL provides a series of guidelines, that will be considered here [45].

- **The number of concepts should be reasonable** - GEMAL is limited to 8 object types (including the unspecified thing), and a low number of relationships and other concepts, which should be well withing reasonable range. In comparison, UML has 233 different concepts. However, in addition to understanding the concepts, participants must understand what each concept means in different contexts, which it will require additional training to comprehend. GEMAL should satisfy Moody's ideas for good graphic economy (GE), stating that the number of different graphical symbols should be cognitively manageable [51].
- **(SEQ-8) The core concepts must be general rather than specified** - GEMAL does well in this regard, the core concepts have few restrictions, and can thus be used to model just about anything.
- **(SEQ-9) The concepts must be composable, which means that we can group related statements in a natural way** - The composite structure of GEMAL lends itself to this. Every specialisation of thing can act as a container for grouping purposes. This guideline is related to Moody's concept of complexity management (CM) [51].
- **(SEQ-10) The language must be flexible in precision** - Models can be built of concepts with different levels of precision. Concepts may be further specified by relationships or sub-concepts, or left unspecified as an empty container. Some level of ambiguity may arise from the interpretation that may be necessary to determine the meaning of sub-concepts in some contexts. GEMAL has no built in way to represent vagueness or uncertainty. Examples of such representations can be found in SeeMe [34].

- **(SEQ-11, SEQ-15) The concepts of the language should be easily distinguishable from each other** - This guideline is related to Moody's concept of perceptual discriminability (PD) [51]. The set of elementary graphical techniques available for constructing visual notations includes horizontal position, vertical position, shape, brightness, size, orientation, colour and texture [10]. As shown in figure 4.1, GEMAL primarily uses shape as the main mechanism to distinguish its object types. Redundant coding is not utilised, that is, combining multiple visual variables to better distinguish between concepts. Perceptual popout and textual differentiation are other techniques for increased perceptual discriminability that are unused. Positioning both horizontally and vertically is used along with relative size to represent compositions. GEMAL does not apply orientation or textures. Both processes and products are represented graphically using a form of rectangle, and may require more effort to be distinguished. The only colour that is used is grey to symbolise instance level concepts. This complies with Moody's argumentation stating that shapes, lines and text are the most commonly used notations [51]. The degree in which the language utilises the full range of visual variables is what Moody calls visual expressiveness (VE).
- **(SEQ-13) The language must be flexible in the level of detail** - Level of detail is controlled by composition structures and inclusion or omission of properties on concepts.
- **Statements must be easily extendible with other statements providing more details** - Again, this seems to be handled well by composition strategies.
- **(SEQ-14) The language must be able to represent the intention of the model** - Intention can be modeled at any level by assigning GEMAL-goal to any other component. Goals can also be decomposed.
- **(SEQ-16) It should be easy to distinguish which symbol a graphical mark is part of** - As shown in fig. 4.2, graphical marks are contained within the symbol that they are a part of. I.e. it should be reasonable to assume that the state symbol belongs to the outmost process in fig. 4.2a.
- **(SEQ-17) The use of symbols should be uniform (one symbol is used for one concept)** - This is true for GEMAL.
- **(SEQ-18) Symbols are as simple as possible** - Symbols (fig. 4.2) appear to be as simple as they can be, while still being distinguishable (SEQ-11, SEQ-15).
- **(SEQ-19) The use of colour is appropriate** - Grey background for instance level concepts seems appropriate. More use of colour could prove efficient.

- **(SEQ-20) The use of emphasis is appropriate** - Size and positioning variables are used to emphasize. Perhaps more variables could be added here.
- **(SEQ-21) Composition of symbols can be made in an aesthetically pleasing way** - Ref. fig. 4.2 and 4.3, it seems possible to create aesthetically pleasing compositions. It remains to prove that this scales up with more complex models.

So far we have not touched on Moody's concept of semantic transparency (ST), which states that visual representations should suggest their meaning [51]. Judging by fig. 4.2, the concept symbols explain themselves in varying ways. The person and organisation symbols may mimic their physical counterparts the best, by appearing like a person and a house/organisation, respectively. However, would the typical model interpreter think that a house is closer related to an organisation than to a location? The location symbol is designed as an open 'cup' to represent something that you can place other things in. Note that locations represented in GEMAL does not need to be tied to a physical point.

Given that possible symbol ambiguity, the person symbol is the only one in the set that can be viewed as semantically immediate (that is, a novice reader can be expected to be able to infer its meaning from its appearance alone) [51]. The symbols for process, product, goal and capability can be seen as semantically opaque, meaning that there is purely an arbitrary relationship between their appearance and their meaning. Advanced readers may be able to derive that the square symbols represent processes and products, given that the same representations are common in other modeling languages. Are the organisation and location symbols semantically perverse, meaning that a novice reader would be likely to infer a different (or even opposite) meaning from their appearance? Empirical research is needed to find conclusive answers.

Krogstie argues that several principles from gestalt psychology, as written by Colin Ware [83], can be applied to conceptual models [45]. This can be used to ground the use of shape, colour, lines and more in psychology. Among other things, it is claimed that closed contour shapes are best suited to represent a concept of some kind in node-link diagrams.

The following discusses GEMAL relative to other concepts of Moody [51] that we have not touched on earlier in the evaluation:

- **Cognitive fit (CF)** - The main idea here is that different visual representations (dialects) may be suitable for different audiences, i.e. looking at expert-novice differences. GEMAL in its current state does only have one set of representations meant to fit every situation.
- **Cognitive integration (CI)** - This category is split into conceptual integration (CI) and perceptual integration (PI). CI seeks mechanisms to help the reader assemble information from separate diagrams into a coherent mental representation of the system. Support for this in GEMAL is exemplified in figures 4.3 and 4.2b; the 'proposal' product is visualised in

two different ways with varying level of detail. Figure 4.3 provides a summary (long shot) view of the system as a whole. GEMAL does not have support for PI at this stage.

- **Dual coding (DC)** - GEMAL does not use a combination of text and other factors to aid perceptual discriminability. It does however utilise separate textual conventions for i.e. processes and products. Processes follow a verb-noun naming convention ('Decide conference location'), while products get a description in noun form ('Proposal').

4.2.3 Participant Appropriateness

Participant appropriateness (PA) relates the participants knowledge to the language. The conceptual basis should correspond as much as possible to the way individuals perceive reality. It is primarily a means to achieve semantic quality (for the modeler) and pragmatic quality (for the model interpreter) [45]. Here we consider the PA guidelines included in SEQUAL. Note that empirical evaluation techniques seem to be needed to be able to provide clear assessments relative to the guidelines.

- **(UEML_25) EM [the language] should be easy to learn, making an easy training** - Assessing this guideline will require empirical evaluation.
- **(SEQ-3) The language should use concepts familiar to the enterprise users** - Since GEMAL is intended for general use by any enterprise, it is a challenge to use concepts that are familiar to everyone. The 7 specialisations of thing should be general enough to be easily understandable by any business user.
- **(SEQ-4) The notation should be intuitive for the enterprise users** - Assessing this guideline will require empirical evaluation.
- **(SEQ-5) It should be possible to easily show the level of completeness and consistency of a part of a model** - GEMAL does not have any dedicated support for this requirement. Closed concepts acting as containers does not reveal how complete or consistent they are, without requiring the participant to open the container and view its details.

4.2.4 Tool Appropriateness

Tool appropriateness relates the language to the interpretation from the technical audience (tools) [45]. For tool interpretation to be possible, the language has to lend itself to automatic reasoning. This requires formality (both formal syntax and semantics being operational and/or logical). Mathematical semantics with high *analysability* or operational semantics with high *executability* are efficiently executed by technical tools.

Tool appropriateness does also include how the language lends itself to manual tools in modeling sessions and modeling conferences, like wall-charts and whiteboards. In these cases, requirements for perceptual discriminability, semantic transparency and visual expressiveness will be different.

GEMAL does obviously not have an operational or mathematical implementation at the point of language proposal, as the work documented in this thesis largely is based around designing and evaluating a technical implementation of the language. We will get back to considerations related to tool appropriateness in a later chapter.

4.3 Summary

It was found that GEMAL lacks representation capability for some of the concepts of the BWV model, mainly on topics including properties and events. Adding the missing concepts may be tempting, but it will be important to keep the range of concepts in the language low. With increased expressiveness comes increased complexity. It may also prove a challenge to incorporate the mentioned concepts in a formal unambiguous way to a holistic modeling environment.

Note that it is not a goal in itself to adhere to every principle and conceptualisation of the BWV model; a model may be logically and practically sound even when it is using a very different set of conceptual components. Identifying how the modeling language compares to an established theoretical basis can however help ground it in literature and understand its focus and mechanics.

Perceptual integration techniques were not discussed in the GEMAL language proposal. Such techniques become more available when the language is implemented in a technical tool. The model user may benefit from visual cues that tell him where in the model he is, where he came from, and where he can go from there. GEMAL-things may also benefit from having a way of showing whether they contain sub-things and if so indicate how many. This way the readers could be guided to inspect composite things should they seem relevant to his interests.

GEMAL does not have notations dedicated to representing incompleteness and vagueness. As discussed in section 2.3.2.4 in the context of the SeeMe language, such notations can prove valuable in certain scenarios. Their applicability to GEMAL should be considered, but again, ensuring good graphical economy by limiting the amount of visual variables is of importance.

GEMAL may benefit from employing a wider range of visual variables. We found that shape and position are the dominant visual variables in GEMALs notation. Experimenting with colour, orientation, icons and textures could improve the languages visual expressiveness and perceptual discriminability.

Chapter 5

Implementation of GEMAL

5.1 Expanding the Language

We will here discuss concretisations, additions and changes done to the GEMAL language, based on the initial evaluation (see section 4.2) and theoretical investigation (chapter 2). The GEMAL language proposal (section 4.1) did not discuss all the practical matter that need to be concretised when implementing such a language in a modeling environment.

5.1.1 Metamodel

The metamodel in fig. 5.1 defines the components included in the most current version of the GEMAL modeling language. It reflect how components relate to each other in the technical implementation in Metis software tools, and also acts as a reference for those seeking knowledge about how to model correctly using GEMAL. Many of the components are carried over from the GEMAL language proposal (ref. chapter 4), but others have been introduced as a part of the implementation. We will go through the components of the metamodel briefly here; they are detailed individually in later sections of this chapter.

- Thing, Decision Point and Relationship are *specialisations of Modeling Object*.
- Capability, Goal, Location, Organisation, Person, Process and Product are *specialisations of Thing* (a thing can be of any of these types).
- Property describes Thing (a thing can have zero to many properties).
- Communication, Generalisation, Govern, Interrupt, Precede, Support, Instantiation are *specialisations of Relationship* (a relationship can be of any of these types).
- Relationships can connect any component that is part of the Modeling Object generalisation structure.

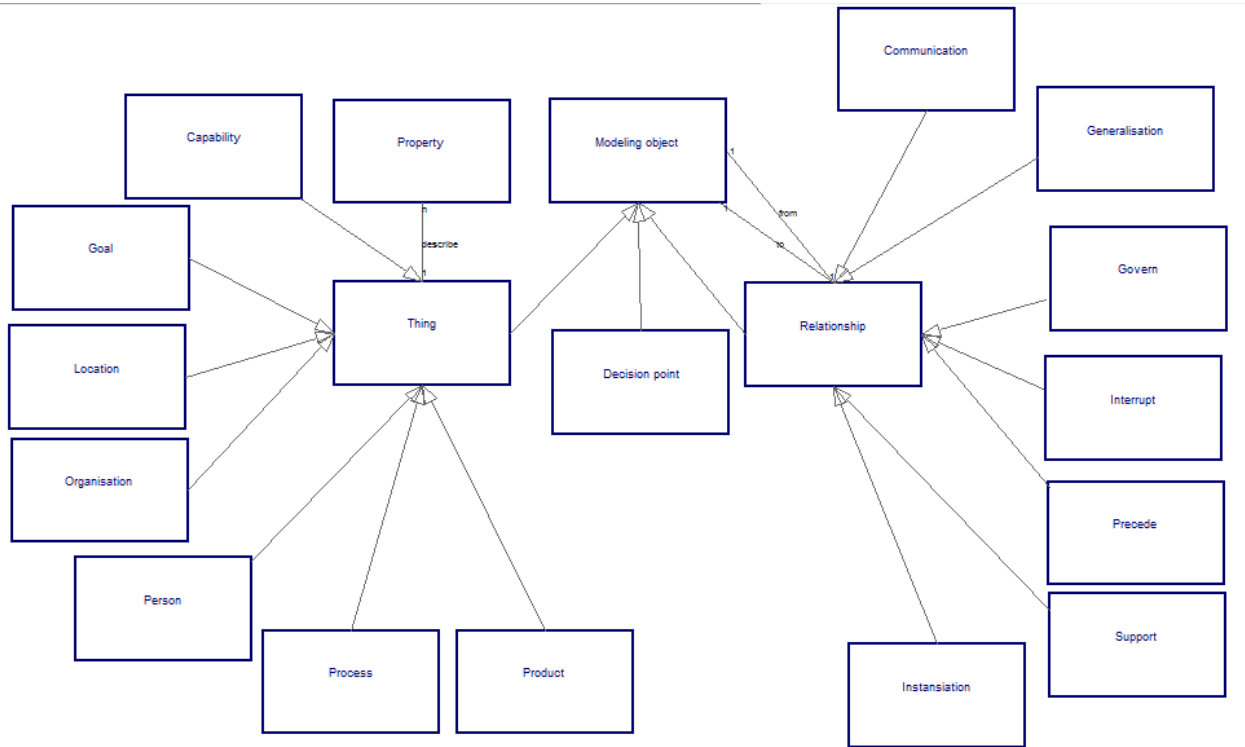


Figure 5.1: GEMAL Metamodel

5.1.2 Things

GEMAL thing and its specialisations (Capability, Goal, Location, Organisation, Person, Process, Product) have not changed much since the language proposal. The initial evaluation of the language did not reveal that any changes to these basic concepts was needed. However, some changes and concretisations related to specific details for GEMAL 'Thing' has been done (e.g. for state, modality, properties, symbols, and visualisation of properties), as is described in later sections of this chapter.

5.1.3 State Space of Things

The GEMAL language proposal states that each (specialisation of) thing should have a set of possible states, but did not define the state space. The following states have been selected, with the goal of covering the lifetime and usage of concepts:

- **Planned:** The thing has gone through (initial) planning. Every aspect of the thing may not yet have been prepared, and an estimated starting time may not yet be available.
- **Waiting:** The thing has been planned, and is now in a passive state. It may be waiting for some date or for the state of another thing to change before becoming ready.

- **Ready:** Requirements that have to be fulfilled before the thing can become active has been met. The thing may be waiting for a signal, for execution resources to be freed, or an active decision by an authorised user.
- **Active:** The thing is currently active. It may be (part of) an active process or activity.
- **Finalised:** Processes or activities related to the thing has ended in the way it was intended to end.
- **Terminated:** Processes or activities related to the thing has ended because it was interrupted (it did not end in the way which was intended)

The states are displayed visually on things in GEMAL models. When the basic (default) view style is active, the state indicator will only show on open things (e.g. its internal parts are shown). When the advanced view style is active, the state indicator will be displayed both on open and closed state things (see sec. 5.1.12 for details on view styles in GEMAL)

Symbols used for possible states in GEMAL can be seen in fig. 5.3b. New with the implementation of GEMAL is the colour sceme used along with the textual notation scheme stemming from the language proposal, as can be seen in the figure. This is an addition made as a part of an effort to increase the visual expressiveness of GEMAL, as detailed in section 5.1.12.

Possible state transitions in GEMAL are shown in fig. 5.2. If everything goes according to plan, a thing should follow this state pattern over its lifetime: (1) Planned, (2) Waiting, (3) Ready, (4) Active, (5) Finalised. The thing can be suspended from (3, 4) setting its state to Waiting (2). It can also be terminated at (2, 3, 4), setting the state to Terminated (6).

The state space of GEMAL is a little different from the state space of other modeling languages. EEML [45] has the following states (for processes): 'not started', 'on scedule', 'behind schedule', 'completed', 'postponed', 'canceled', 'non-working'. BPMN has 'start' and 'end' events in place of start/end states, that can be modified with various notations [70]. In addition BPMN has a series of events (some categories are 'message', 'timer', 'escalation', 'error', 'cancel', 'compensation', 'signal') that parallel some of the representation of GEMAL states. One could easily imagine more states that could have been added to GEMAL (i.e. 'failed', 'escalated'), but as part of an effort in retaining GEMAL as a simple language with few concepts, they have not been included at this point. The selected state space is meant to provide simple means of indicating the state of a thing.

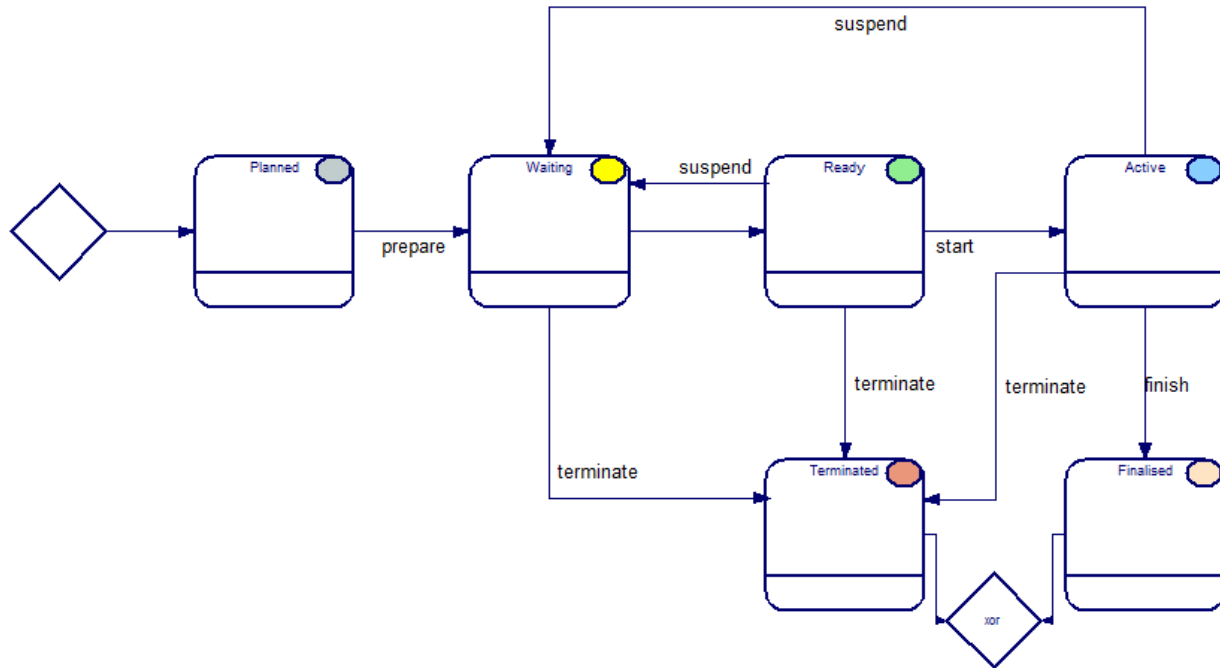


Figure 5.2: Possible state transitions in GEMAL

5.1.4 Modality Space of Things

No changes have been made to the available modality values for GEMAL things since the language proposal. However, the language proposal did not list what each value means in the context of GEMAL thing, so that will be outlined here:

Deontic Modality (linguistic modalities that indicate how the world ought to be, according to certain norms, expectations, speaker desire, etc.)

- **Obligation:** A thing with this modality value contains requirements which must be fulfilled.
- **Recommendation:** A thing with this modality value contains a recommendation (typically a course of action) for other thing(s).
- **Permission:** A thing with this modality value grants authorization to do something (a formal consent) for other thing(s).
- **Discouragement:** A thing with this modality value discourages some behaviour or action for other thing(s).
- **Prohibition:** The thing with this modality value prohibits some behaviour or action for other thing(s).

Alethic Modality (linguistic modalities of truth, in particular the modalities of logical necessity, possibility or impossibility)

- **Necessity:** A thing with this modality value can contain something that is a required condition for something else to be the case in other thing(s).
- **Contradiction:** A thing with this modality value opposes or denies some behaviour or action for other thing(s).

Each thing with a modality value will have one or more recipient things. Which thing(s) are affected will depend on the context of the thing with the property. If the thing is part of a decomposition structure, the parent would typically be affected by the modality. The thing could also be related to another thing via the 'Govern', 'Support', or 'Communicate' relationships, to denote which thing(s) are affected by the modality. Modality can be visualised on GEMAL things by enabling the advanced view style (see sec. 5.1.12 for details).

5.1.5 Introducing Complexity and Vagueness

Along with the implementation of GEMAL, an effort to increase Cognitive Fit, Perceptual Integration, and Visual Expressiveness while not impacting Graphical Economy, Perceptual Discriminability, and Complexity Management too much negatively (c.f. Moody's notation in section 3.2.2.2) of GEMAL was made. This implied making it possible to add information that can be beneficial to interpreters of GEMAL models, and keeping several things in mind while doing so: (i) The notation should remain cognitively manageable for both novice and advanced users (CF), (ii) be careful about increasing the number of concepts (GE), (iii) current concepts should not become overly complex (PD).

A notation for expressing the internal complexity of GEMAL Thing has been added. An example is given in figure 5.7. Both its sub-figures represent the same GEMAL 'Product', but figure 5.7b has the advanced view style enabled, which makes more information display visually. The number '2' in leftmost square in the bottom of the figure indicates that the Product has a complexity level of two, meaning that one additional component can be found decomposed inside it. This functionality aims to better the Perceptual Integration and Visual Expressiveness of GEMAL models. A model interpreter may find himself inspecting a model with a given level of abstraction. The complexity indicator will let him or her know the number of internal sub-elements of the things in the view, and thus help draw the person's attention to complex modeling constructs that may be of interest.

Inspired by the SeeMe modeling notation (see section 2.3.2.4), GEMAL has received a visual notation able to indicate incompleteness and vagueness of concepts. The middle square in the bottom part of figure 5.7b shows an example of this. The SeeMe (see figure 2.7) notation separates three types of incompleteness; (i) intentional incompleteness (blank box), (ii) More

knowledge needed (three dots), (iii) doubtful about correctness (questionmark). Inspired by these types, the following vagueness indicators have been added to GEMAL:

1. **Must Validate:** The correctness of the component needs to be further validated (there is doubt about its correctness).
2. **Missing by Purpose:** Details have been intentionally left out of the component (intentional incompleteness).
3. **Missing by lack of knowledge:** More knowledge is needed before the component can be finished.
4. **Complete:** Modeling of the component complete, there has been stated no doubt about its correctness and completeness.

The vagueness indicator of a thing can also be left blank, simply meaning that no comment about the thing's completeness has been made available.

5.1.6 Concretised Relationships

The GEMAL language proposal listed the following relationship types: general, precedes, supports, communicates, governs. Figure 5.3a shows a visualisation of the mentioned types, combined with some types added with the implementation in an effort to increase the language's expressiveness.

This first set of relationships were mentioned in the language proposal, and have been carried over to the implementation (numers correspond with the visual representation of the relationship, as shown in figure 5.3a):

1. **Communicate** - A thing communicates with (informs) another thing.
3. **Generic relationship** - Used for generic (undecided) relationships between things. Renamed from 'general relationship' in the language proposal, as this would create a naming conflict in combination with the new relationship type 'generalisation' (see below).
4. **Govern** - A thing restricts and influences the state of another thing.
7. **Precedes** - Used to connect a series of things that precede each other in time. Useful for process flow diagrams.
8. **Support** - A thing can support another thing in achieving a state change or goal.

The following relationship types were not included in the language proposal, but have been added to the implementation with the goal of providing increased expressiveness to GEMAL:

2. **Generalisation** - Can be used to build generalisation hierarchies, e.g. create relationships between a thing and its sub-things.
5. **Instance of** - A thing can be an instance of another thing. This can i.e. be used to show that a car is the instance of a model.
6. **Interrupt** - Can illustrate that a thing interrupts a process (work, transformation) of another thing. This type was inspired by the added expressiveness a similar type provides in the BPMN language.

Relationships can connect any two components that are part of the Modeling Object generalisation structure of GEMAL (c.f. fig. 5.1), and may also connect GEMAL-Property.

5.1.6.1 Relationship Visualisation Scheme

The relationship visualisation scheme strives to make it easy to distinguish different relationship types, while lending itself to user interpretation - by using conventions that are often found in popular notations. Dotted lines are typically used to symbolise communication (i.e. UML, EEML, BPMN), and an empty arrow-head commonly represents generalisation (in i.e. UML). The 'Governs' relationship utilises a visual variable from the thickness category.

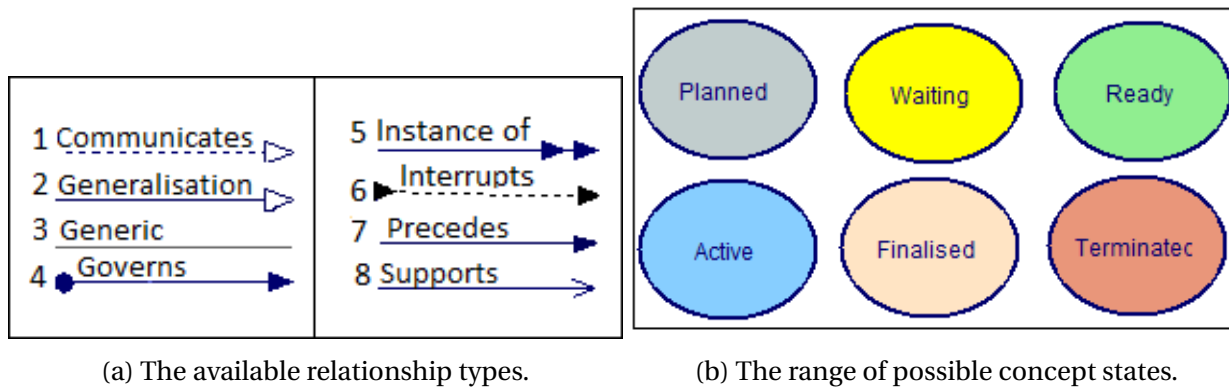


Figure 5.3: Updated visualisation in the GEMAL implementation

5.1.7 Logical Gates

As mentioned in section 4.1.2, logical gates were intended as a part of GEMAL from the language proposal. We have decided to include three types; AND, OR, XOR. See their visualisation in figure 5.4. Logical gates are simple objects with no other properties than name, description, and logical relation (which can hold the values " " (default OR), and (AND), xor (XOR)). Alternatively, logical gates could have been introduced to GEMAL as a specialisation of thing alongside 'Product',

'Person' and the five other current types. This would make it possible to describe the context of a decision point in terms of other 'things'. However, it was chosen to create logical gates as a concept separate from 'thing', to keep them simple and familiar to users that have experience with modeling languages that also treat them as simple concepts.

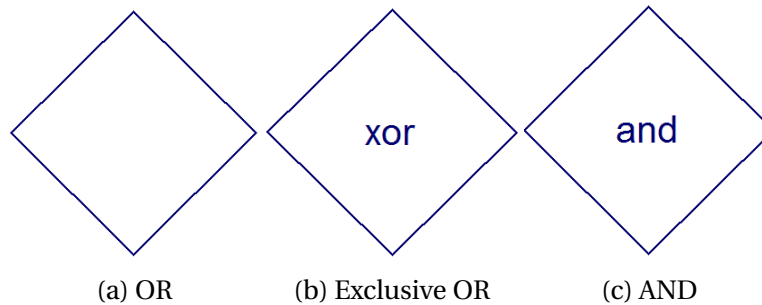


Figure 5.4: Three types of logical gates (decision points) included in GEMAL.

5.1.8 Instantiation Level

As described in section 4.1, an instance/type differentiation was already outlined for GEMAL in its state as a language proposal (type level things are given a white background, while instance level things get a grey background). A problem with this technique is that one may encounter modeling scenarios where concepts in a model may alter between being a type and an instance depending on the current context of use, or the role of the actor currently viewing the mode.

To alleviate this issue, the 'instance of' relationship type has been introduced to GEMAL (c.f. sec. 5.1.6). This does reduce the language's Semiotic Clarity by introducing construct redundancy, which the language was found to have none of at the language proposal stage (see section 4.2.1). While Wand and Weber suggests that modeling languages should strive to avoid CR (as discussed in section 4.2.1) [80], the additional construct may prove beneficial in certain modelling scenarios, as exemplified by the following example:

Volvo, Audi and BMW are car brands. S80, V70 and XC90 are some of the car series available from Volvo. Each individual S80 car belongs to a model (i.e. S80L, T5), and has a set of configurations. Person A is interested in car brands and thinks of Volvo, Audi and BMW as instances of the type 'Car'. Person B is interested in cars from Volvo, thinking of Volvo as the type and S80, V70 and XC90 as some of its instances.

To make a GEMAL model illustrating the relationship between brands, series and models of cars that matches the context and viewpoint of both person A and B, it would not be sufficient to use white background for types and grey background for instances (this colour scheme is exemplified in figure 4.2). Person A's context requires a white background for 'Car' and grey for 'Volvo', while

person B's requires a white background for 'Volvo' and grey background for 'S80'. The 'instance of' relationship type elevates this problem by changing up the representation of types and instances. Both person A and B are happy with the model saying 'Volvo is an instance of car, and S80 is an instance of Volvo'. They no longer have to argue about whether Volvo *or* S80 should be the instance level object.

5.1.9 Expressing Events in GEMAL

As covered in section 2.1.2.1, BPMN is an example of a modeling language in which events are a core concept. Event types of BPMN 2.0 core are 'None', 'Message', 'Timer', and 'Error'. The types can be 'Start', 'Intermediate', and 'End' events (see fig. 2.2 for a more complete overview). Events were also discussed in relation to the BWW model in section 4.2.1.

As a part of the implementation of GEMAL, whether the language could benefit from having events implemented was considered. It has been decided against introducing events as a dedicated concept, since it can be argued that the functionality of the GEMAL 'Precede' relationship in the context of GEMAL 'Thing' overlaps with it. As mentioned in section 5.1.6, 'Precede' can be used to connect a series of things that precede each other in time. This implies that precede can connect a thing of specialisation 'Process' or 'Decision Point' with name 'Start' to reproduce a BPMN 'Start' event. A 'Process' 'Decision Point' can be given a Start-time to reflect a BPMN 'Start' event, or an end time for 'End' events. This can be applied in concert with the GEMAL 'Communicate', 'Support', and 'Interrupt' relationships to reproduce other BPMN events.

5.1.10 Properties of Things

A set of properties is available to GEMAL things. Every specialisation of thing (i.e. 'Person', 'Process') can have the same properties. The following list of properties correspond to the display name of the properties in the Metis implementation of GEMAL:

- **Name:** The name of the thing (i.e. 'Bob' or 'Person A' for a person). Will display visually on the concept in form of text.
- **Description:** A description of the thing with optional length. Will only display when the thing is inspected.
- **Start-time:** Time and date for when the thing starts. Will only display when the thing is inspected. The meaning of 'starting' may depend on the nature of the thing, and can be interpreted differently depending on the context (i.e. a person was born or entered the company when he started, a location was built or discovered when it started).
- **End-time:** Time and date for when the thing ends. Will only display when the thing is inspected. The meaning of 'ending' may depend on the nature of the thing, and is not

necessarily a meaningful property in every case (i.e. what does it mean that the location 'My School' ends?).

- **Instantiation Level:** Drop-down list where the available choices are 'type' or 'instance'. The thing will get a grey background when 'instance' is selected.
- **Modality:** Drop-down list where the available choices are 'necessity', 'obligation', 'recommendation', 'permission', 'discouragement', 'prohibition' and 'contradiction'. The modality value displays visually on closed things when the advanced view style is active.
- **Current State:** Drop-down list where the available choices are 'Planned', 'Waiting', 'Ready', 'Active', 'Finalised' and 'Terminated'. These values match the GEMAL state space described in section 5.1.3. Current state will be displayed visually on closed things when the advanced view style is active. C.f. visualisation in fig. 5.3b.
- **Vagueness:** Drop-down list where the available choices are 'Must Validate', 'Missing by Purpose', 'Missing by Lack of Knowledge', 'Complete'. These values match the vagueness state space described in section 5.1.5. The vagueness value displays visually on closed things when the advanced view style is active.
- **Reference:** A text string that can link a GEMAL thing to an external resource. URLs can point to resources on the web, while file paths can point to a file on the users computer. A web resource will utilise the users preferred browser, while local files will be opened in a program determined by the type of file referenced (i.e. .doc will use Microsoft Word).

In addition to the mentioned default properties for GEMAL things, a new concept called 'Property' has been introduced to GEMAL (see figure 5.5). 'Property' is intended to be used when the goal is to add a easily visible property to a thing, and the modeler is unable to categorise the property using one of the preconfigured specialisations of thing (i.e. 'Location', 'Capability'). A property of this form may contain other properties, making it possible to create structures for more complex properties. However, 'Property', similar to logical gates (see justification in section 5.1.7), has been implemented in a simple form. As such, 'Property' can not contain 'Thing', logical gates, or relationships. Properties can be copied to other things, thus creating a mutual (shared) property, where both instances will change when one is edited.

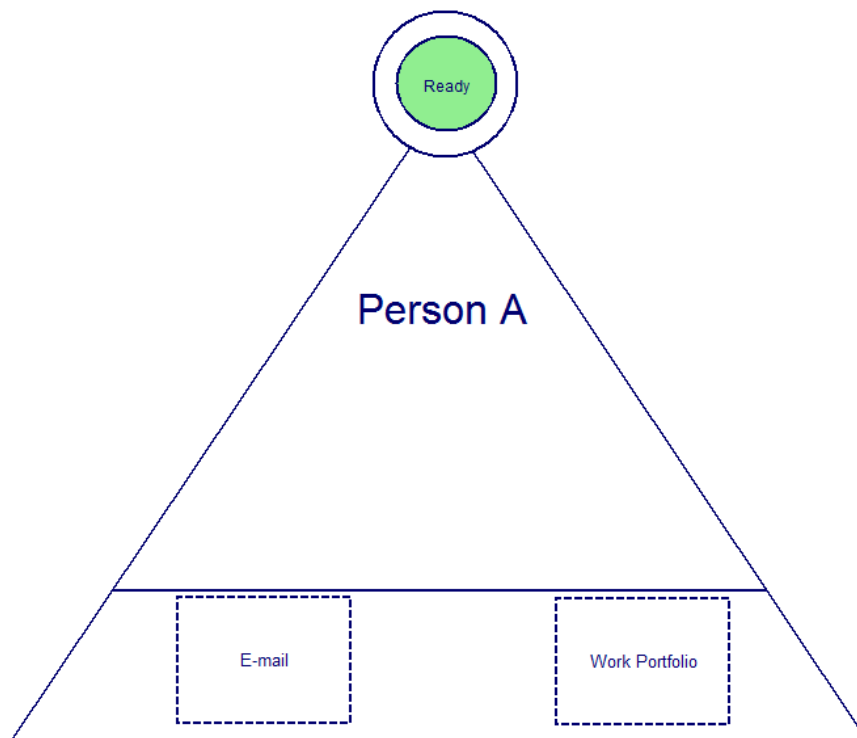


Figure 5.5: An example person with two 'Property' objects added to it. 'Email' may have its 'Open Document' method set to opening an editor where the model interpreter can send an email to 'Person A', while 'Work Portfolio' could link to a web-source containing the person's recent work.

5.1.11 Changes to Symbols

As a part of implementing the GEMAL language in Metis, symbols introduced through the GEMAL language proposal were evaluated (c.f. sections 7.1 and 7.2). However, symbols for GEMAL concepts were implemented prior to these evaluations.

The symbol for the GEMAL location concept has the only symbol that received a small visual update, making it a closed contour symbol. See fig. 5.6. We briefly discussed the use of closed contour shapes for language concepts in section 4.2.2. No indications for bad visualisation of other GEMAL concepts were found in the initial evaluation, so no other symbols have changed for the implementation.

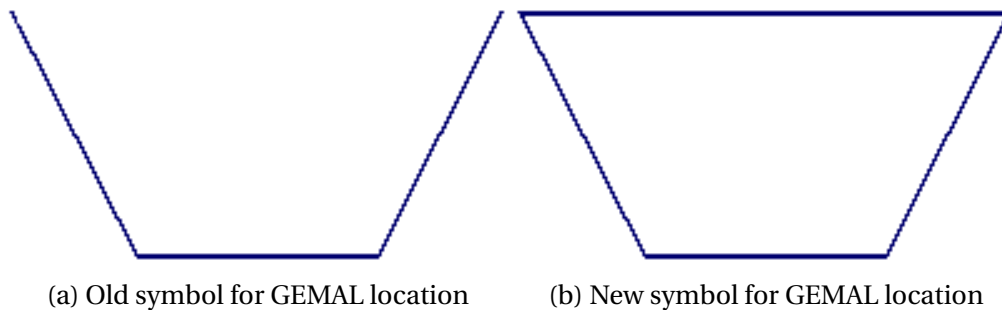


Figure 5.6: Updated symbol for GEMALs location concept

5.1.12 View Styles: Optional Increase to Visual Expressiveness

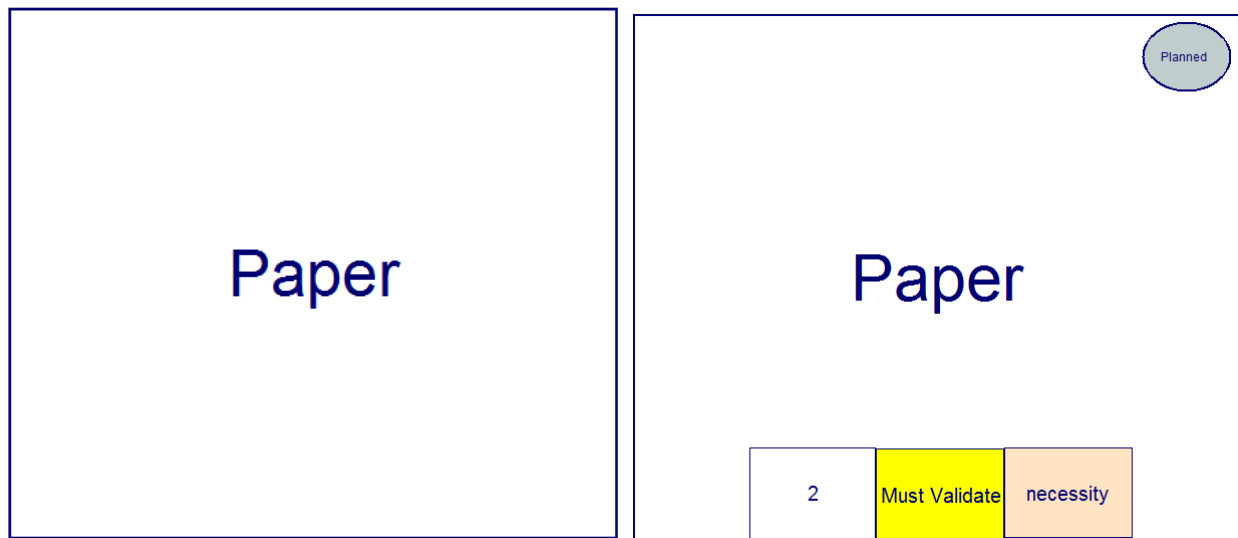
As mentioned in sections 5.1.4 and 5.1.5, GEMAL's visualisation has been divided into two view styles in the implementation. The advanced view style has all the visualisation of the basic view style, with a few additions; variables for state, complexity, vagueness, and modality for a thing are displayed graphically on the thing when it is closed. State is the only one of the mentioned variables that is displayed at all in the basic view style, and then only on opened things. See fig. 5.7 for a comparison of the two available view styles on closed GEMAL concepts.

The variables introduced for complexity, vagueness, and modality aim to better the cognitive fit of GEMAL, as users may select a view style depending on their experience, skill level, and needs, and in that way customise the visual expressiveness of the models they are inspecting. The variables are introduced visually in a dual coding effort (text and colour is combined). Moody claims that dual coding is a cognitively effective visual technique that should be used more in graphical encoding. The two different visual dialects will hopefully prove useful for different tasks and to different audiences. In the case of the GEMAL implementation in Metis, the representational medium for models will primarily be a computer screen, so the view style containing colour schemes should not be problematic for users of the Metis implementation (since it is maintained automatically by the system). If GEMAL models are to be transferred to other

mediums (i.e. a whiteboard), users can choose to resort to the basic GEMAL view style. The variables are also intended to maintain solid complexity management for GEMAL, by introducing new information sources without overloading the human mind.

Colours associated with the visualisation of 'State' can be seen in fig. 5.3b. In a similar fashion, possible Modality values has been given the following colouring scheme: 'None' (white), 'Necessity'(bisque), Obligation(yellow), Recommendation(light green), Permission(sky blue), Discouragement(orange), Prohibition(red), and Contradiction(dark salmon). It has been strived to give dominant colours that can be associated with restriction and danger to modality values that are the most important and dominant, while values that are lighter on the rules, such as recommendations, have been assigned more modest colours, such as light green. The colour scheme for the vagueness visualisation employs the following colouring scheme: 'None'(white), 'Must Validate'(yellow), 'Missing by Purpose'(bisque), 'Missing by Lack of Knowledge' (sky blue), and 'Complete'(green). In the case of vagueness, the colours are meant to symbolise the completeness grading of a thing. (Yellow typically means waiting, while green can mean ready/go). Also, in general, colours in the visualisation of GEMAL are intended as a means for experienced users to find things of particular interest for his/her current context, at a glance (cognitive fit).

Increased visual expressiveness for relationship types (as mentioned in sec. 5.1.6) has been implemented for both view styles.



(a) A closed 'Product' with GEMAL's basic view style enabled (b) A closed 'Product' with GEMAL's advanced view style enabled

Figure 5.7: Example of the difference between basic and advanced view style for GEMAL things. Note that both sub-figures represent the same 'Product'.

5.1.13 Other Additions to GEMAL

Here we briefly outline other functionality additions to GEMAL that came as a part of the implementation in Metis tools. These functionalities have not been made from the ground up for this project, but have been imported from the default set of Metis tools.

5.1.13.1 Layout Strategies

Layout strategies are tools that can be used when configuring the layout of a GEMAL model. The included strategies are:

- **Standard Matrix Layout Strategy:** Standard matrix layout.
- **Compact Matrix Layout:** Compact matrix layout, less space between model objects.
- **Presentation matrix layout:** Matrix layout geared towards the model presentation format.
- **Minimum spacing and complete channel merging:** Matrix layout where spacing is minimized and channels are merged where possible.
- **Minimum spacing and sorted objects:** In addition to minimum spacing (like above), objects are sorted alphabetically by name.
- **Process Layout:** The standard process modeling layout. Objects appear smaller than when other layouts are applied.
- **Position-based Layout:** Matrix layout where relative object position is the deciding factor.
- **Standard Matrix Layout. No Relationships:** Like the standard matrix layout, but best used when no relationships are present among the affected model objects.

5.1.13.2 Other functionality

- **Relationship Matrix Button:** Used for storing user-defined relationship matrices. The relationship matrix can be created using the Metis relationship matrix editor.
- **Print Settings Button:** Used for storing printing information, which includes the current model view, zoom level and pan.
- **Image:** Used for inserting an image somewhere in a model. Jpg and png formats are supported.
- **Picture:** Used for annotating a container or the model view background with a symbol. Available symbols can be found in the Metis symbol tree.
- **Shortcut:** Used for starting an (external) Metis model from the current model. Useful when seeking to reference a related model.
- **Pushpin:** Used for 'posting' notes to a model. Visibility of the note can be toggled by clicking the pushpin.

- **Comment:** Used for adding comments to a model or any of its objects. More visible than pushpin notes.
- **Action Button:** Used for performing various actions in a model. Commonly used to zoom to an area in the current model or a related models.

Refer to appendix [B](#) for technical implementation details that have been omitted from the main part of this report.

Chapter 6

Case Studies

This chapter describes the case studies that have been carried out for GEMAL after its implementation in Metis. Models were created in modeling languages including DFD, ER, i*, ArchiMate, and UML (Class Diagram, Use Case Diagram, Activity Diagram). The modeling languages were selected on the basis of the fact that they are extensively used in the industry today, and that they can be classified into different modeling perspectives (c.f. sec. 2.1.2). The idea is that if it can be proven that GEMAL can handle basic models in these languages, we are one step closer to suggesting GEMAL as a viable approach as a general purpose enterprise modeling language. Additionally, a case study intended to explore whether GEMAL can handle large and complicated modeling tasks was conducted (Case 6).

6.1 Case 1: Comparing GEMAL to Data Flow Diagram

The DFD (fig. 6.1a) below is based on the Order Management System case description found in appendix C.1. This particular model shows how the system handles an order from a customer. Order information is passed to the system, where the order is processed. Databases are updated before the ordered goods are shipped and a receipt is issued back to the customer.

Fig. 6.1b depicts a GEMAL version of the DFD in fig. 6.1a. Note that 'Order Information' and 'Receipt', that could be found as relationship labels in the original DFD, have been changed to dedicated 'Product' instances in the GEMAL model. This change was made so that properties could be added to them. This means that while both models show as much complexity on the surface, the GEMAL model can be further inspected to reveal more details not present in the DFD.

For the models in figure 6.1, the 'Advanced' GEMAL view style was enabled.

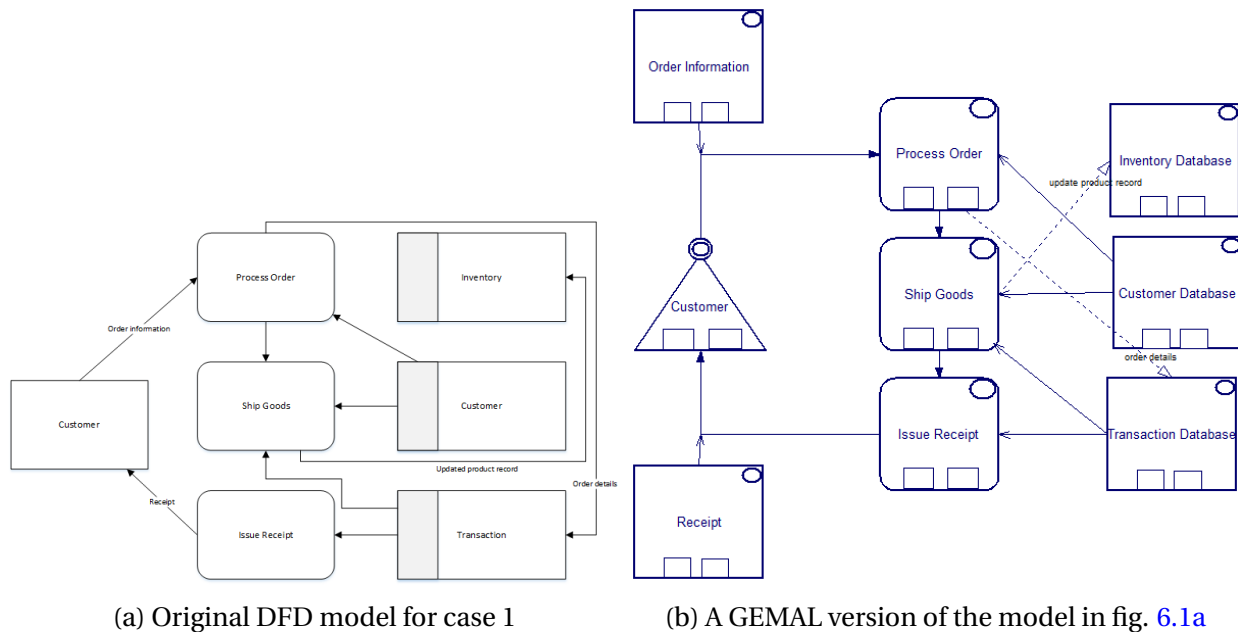


Figure 6.1: A DFD based on the Order Management System case description (see appendix C.1) and a GEMAL version of the same model

6.2 Case 2: Comparing GEMAL to Entity Relationship

The ER model in figure 6.2a is based on the Order Management System case description found in appendix C.1. This particular model shows how customers, products, manufacturers, accounts and orders relate in the system, in addition to showing the properties of each entity.

Fig. 6.2b shows a GEMAL version of the ER model. At first glance the GEMAL model may look less complex. This is because the number of entities shown on this abstraction level is much lower. Relationships are no longer represented by a rhombus shape in the model, they have instead been merged with the lines connecting entities (these lines are called relationships in GEMAL). Properties are not visible in this level of abstraction. They can instead be viewed by opening the available GEMAL things (see fig. 6.3). F. Moody states that hierarchy is one of the most effective ways of organising complexity for human comprehension, as it allows systems to be represented at different levels of detail, with complexity manageable at each level (see principle 4 in Moody's principles for designing cognitively effective visual notations, as outlined in section 3.2.2.2).

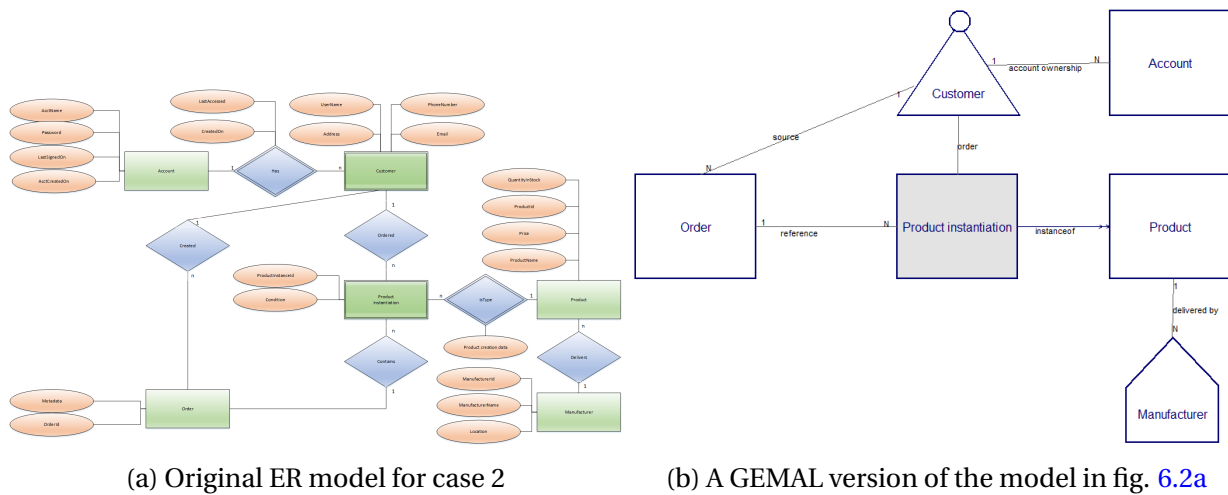


Figure 6.2: An ER model based on the Order Management System case description (see appendix C.1) and a GEMAL version of the same model

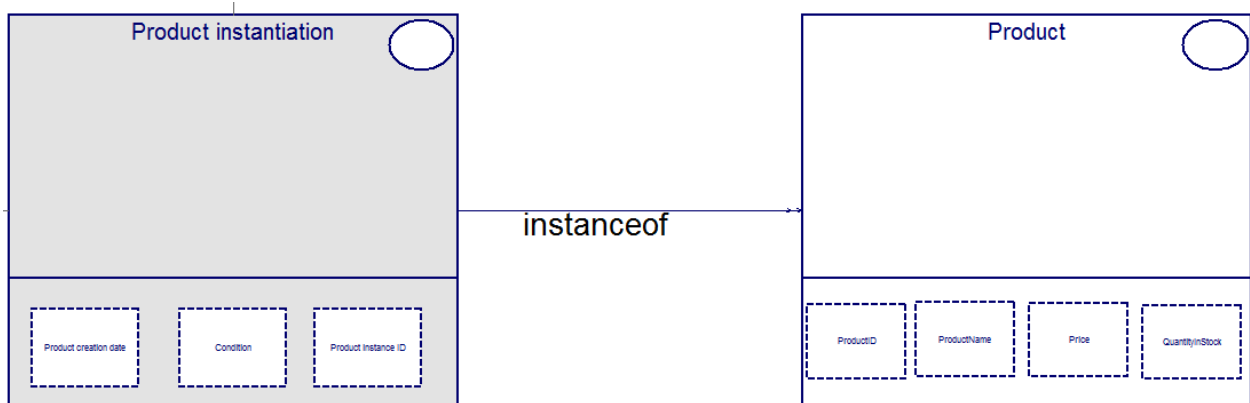


Figure 6.3: A closer inspection of the GEMAL model in fig. 6.2b. Two entities have been zoomed to and opened, revealing their internal properties

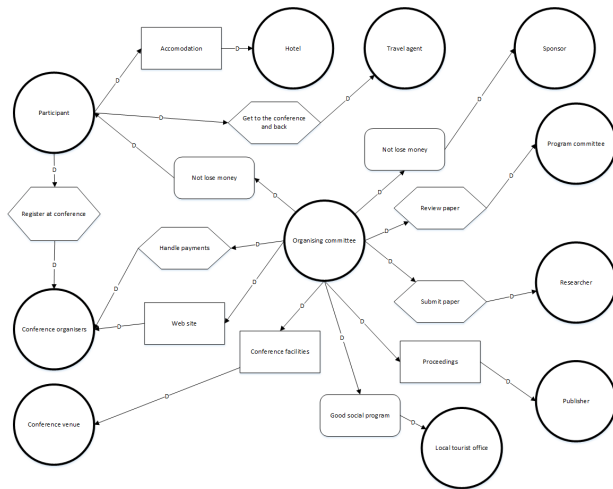
6.3 Case 3: Comparing GEMAL to i*

The i* model of figure 6.4a is based on the 'Arrangement of International Conferences and Events' case description found in appendix C.2. The model shows how different actors relate to tasks, goals and resources when a specific conference is to be arranged.

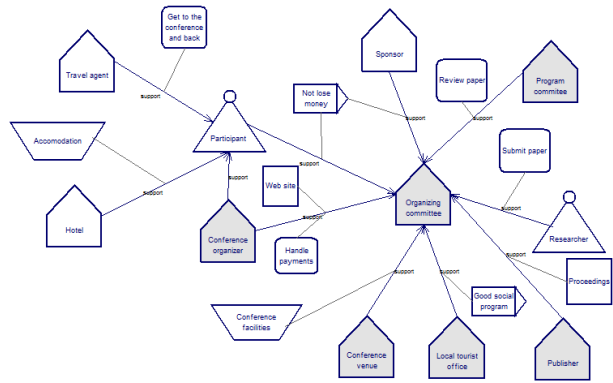
Fig. 6.4b and fig. 6.4c are both versions of the original i* model using GEMAL. Fig. 6.4b may be the one that is visually closest to the original. The same hierarchial structure can be found, but in place of i* 'Actor', different GEMAL things are used ('Organisation', 'Person'). GEMAL 'Product', 'Location' and 'Process' are used in place of i* 'Task' and 'Resource'. i* and GEMAL has similar uses of 'Goal' (this particular model does not contain i* 'Softgoal', which would be represented using GEMAL 'Goal' in typical cases).

Fig. 6.4c is more different from the original. Here, what was 'Goal', 'Resource' and 'Task' in the i* original has now been put in a hierarcially lower abstraction level inside GEMAL things (that in turn represent i* 'Actor' instances). Fig. 6.4d contains a zoomed view of the most complex GEMAL 'Organisation' of fig. 6.4c, where its internal parts are available for closer inspection.

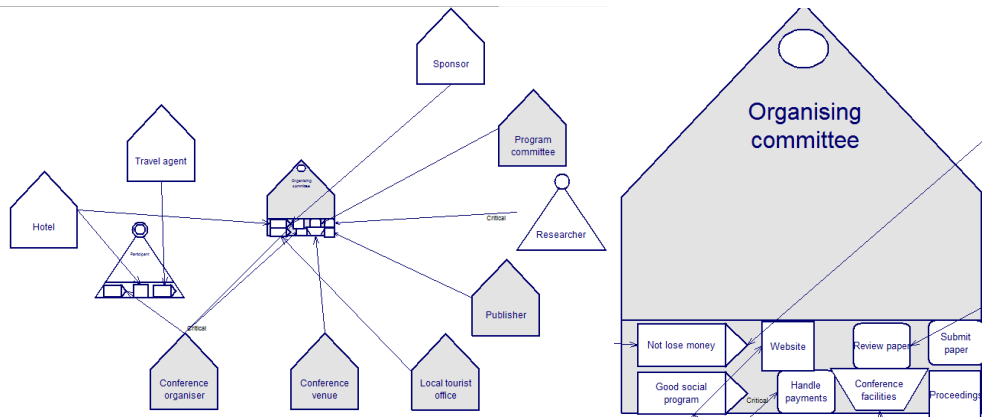
All three models, while different in appearance, contain the same information.



(a) Original *i** model for case 3



(b) A GEMAL version of the model in fig. 6.4a



(c) A second GEMAL version of the model in fig. 6.4a

(d) A zoomed view from the model in fig. 6.4c

Figure 6.4: An *i** model based on the 'Arrangement of International Conferences and Events' case description (see appendix C.2), and GEMAL versions of the model

6.4 Case 4: Comparing GEMAL to ArchiMate

An ArchiMate model based on the 'Order Management System' case description (see appendix C.1) is shown in figure 6.5a. The model shows how in-house systems work in tandem with third-party systems when goods are ordered. The model details the order process as well.

Fig. 6.5b shows a GEMAL version of the ArchiMate model. The models are intentionally quite similar in appearance. However, GEMAL does not have ways to represent every graphical symbol of the ArchiMate original with its own graphical symbols. To combat this, the GEMAL model is using textual differentiation to make it possible to tell the different concepts apart. Where one could tell that the 'Inventory System' component is an application component from the use of ArchiMate 'Application Component' symbol, the GEMAL equivalent has been given the label '«Application» Inventory System' in order to convey the same information. The GEMAL component is classified as a 'Product'.

We can also note that the GEMAL model lacks the option to use colour as a visual variable, making the GEMAL model less visually expressive, albeit with a more managed graphical complexity, than the original.

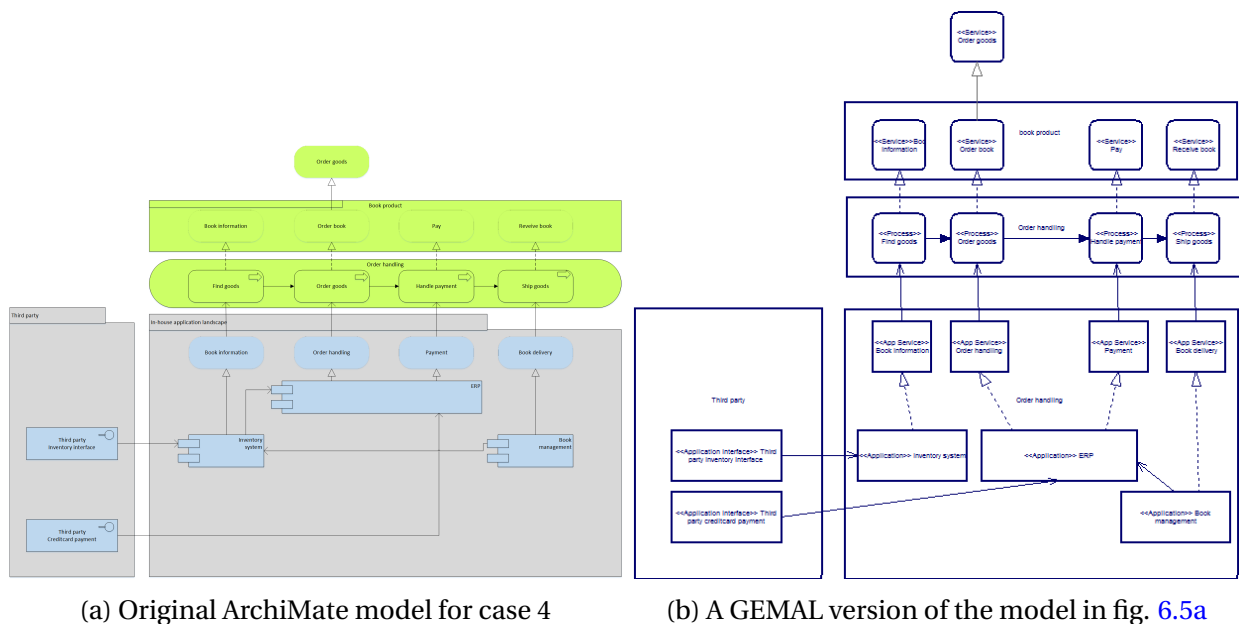


Figure 6.5: An ArchiMate model based on the Order Management System case description (see appendix C.1) and a GEMAL version of the same model

6.5 Case 5: Comparing GEMAL to Unified Modeling Language

UML 2.0 contains 15 different diagram types. In order to create a manageable case study for comparing the sarge UML language to GEMAL, three much used diagram types were chosen.

UML diagrams created based on the 'Order Management System' case description (see appendix C.1) are shown in fig. 6.6. Here follows a short summary of what each diagram displays:

- **UML Class Diagram** (fig. 6.6a): Shows how the classes 'Customer', 'PreferredCustomer', 'ShoppingCart', 'CreditCard', 'ItemToPurchase', and 'Product' relate logically in the system. The cardinality of each relationship is shown on the connecting lines (i.e. '0..* 1'), and variables and functions are displayed inside the respective classes.
- **UML Use Case Diagram** (fig. 6.6b): Shows how different actors (i.e. 'Registered User' or 'Identity Provider') relate to use cases (i.e. 'View Items' or 'Make Purchase' in the web shop). Use cases are part of the 'Online Shopping' sub-system, and relations between use cases are annotated.
- **UML Activity Diagram** (fig. 6.6c): Shows how products ordered through the system are handled with regard to filling the order and sending an invoice to the customer. It displays 'Fill Order' and 'Send Invoice' as the first line of activities after a branch, which are joined together towards the end of the diagram in the 'Close Order' activity.

The GEMAL counterparts of these diagrams are displayed in fig. 6.7. Here follows some notes on differences and similarities found when comparing UML and GEMAL diagrams:

- **GEMAL Class Diagram** (fig. 6.7a): In this GEMAL diagram, 'Customer' and 'Preferred Customer' have been assigned the 'Person' concept. Variables and functions are not visible in fig. 6.7a, to view them the container things at the outmost abstraction level must be opened. Relationships are shown in the same way as in fig. 6.6a. Note that the naming scheme for classes, variables and functions have changed. This is purely a personal preference; one could choose to keep the original UML naming scheme used in fig. 6.6a.
- **GEMAL Use Case Diagram** (fig. 6.7b): Not very different from the UML counterpart. A GEMAL 'Thing' has replaced the UML sub-system. GEMAL 'Person' is used to represent actors.
- **GEMAL Activity Diagram** (fig. 6.7c): Unlike UML, GEMAL does not have dedicated constructs for 'Start', 'Fork', 'Join' and 'End'. Re-sized instances of GEMAL 'Thing' have been used in place of 'Start' and 'End', while GEMAL 'Logical Gate' (the AND gate-type) is used in place of 'Fork' and 'Join'.

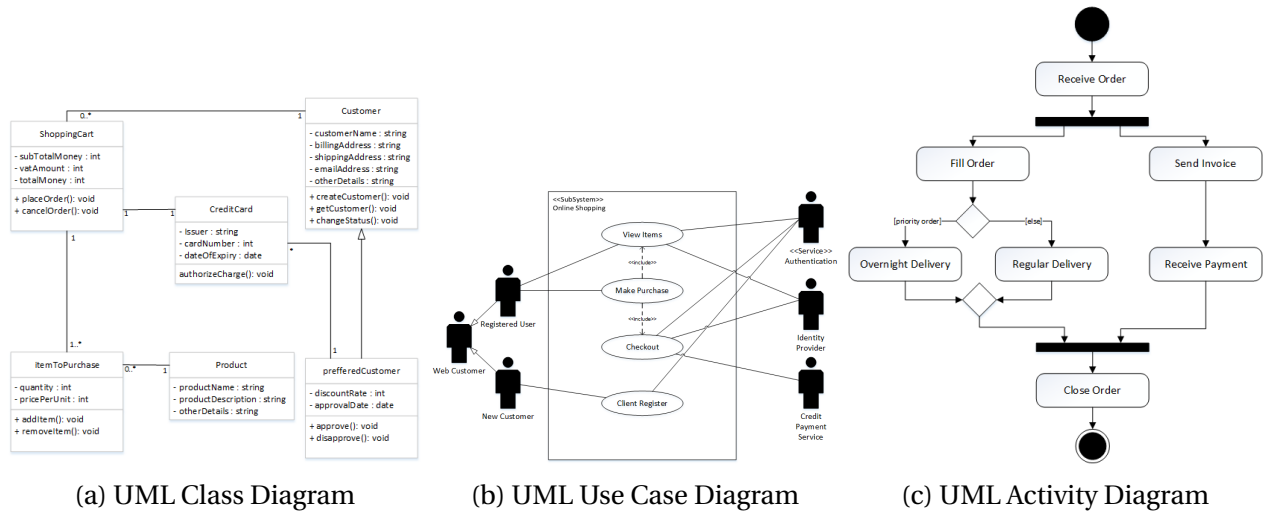


Figure 6.6: Original UML Diagrams used in case 5

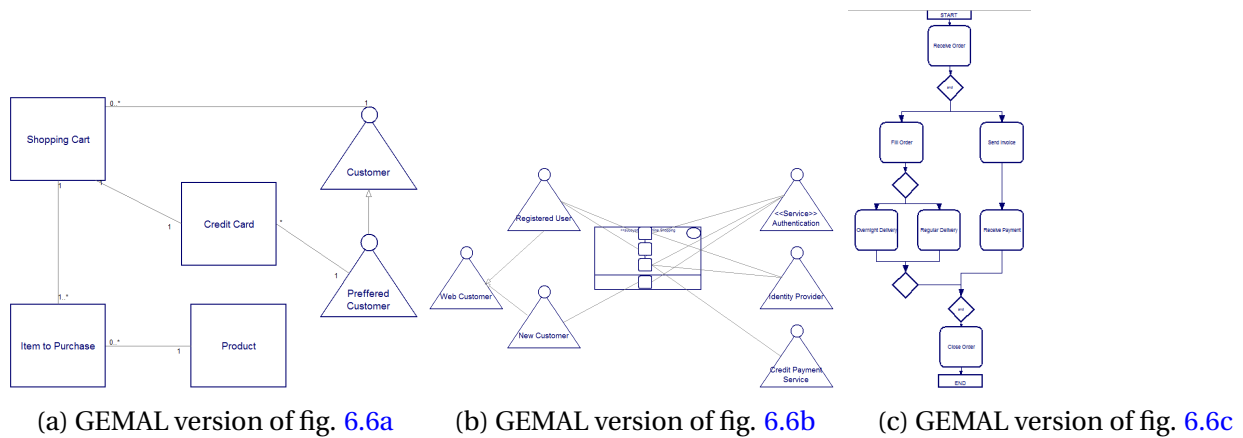


Figure 6.7: GEMAL version of the UML Diagrams in fig. 6.6

6.6 Case 6: Arranging a Conference (Large-Scale GEMAL Model)

This case study was carried out with the goal of finding whether GEMAL can cover complex large-scale modeling tasks. The models (as shown in fig. 6.8) was created by supervisor of this project and author of the GEMAL language proposal, John Krogstie. Please note that the model was implemented in an earlier revision of the GEMAL language, and have not since been updated for the version that is delivered as a part of this project. This means that some notations that were added later on are unavailable (i.e. necessity and modality cannot be visualised in this model). The model, while more complex than any GEMAL model that has been shown so far in this report, is not necessarily complete; it contains examples of how things can be handled, and does not cover every single component and composition that can be of interest in the context of organising a conference. The model can still be opened and viewed by users that have the latest GEMAL version installed.

Fig. 6.8a shows the outer level of a generic conference model. GEMAL 'Things' acting as containers are visible. The things are closed, so that the view is focused on their name attribute. Fig. 6.8b contains the same view of the model, but here the things of fig. 6.8a have been opened, thus revealing their internal decomposition structure. The figure reveals that the things have a greater complexity than what has been the case in other case studies described in this chapter.

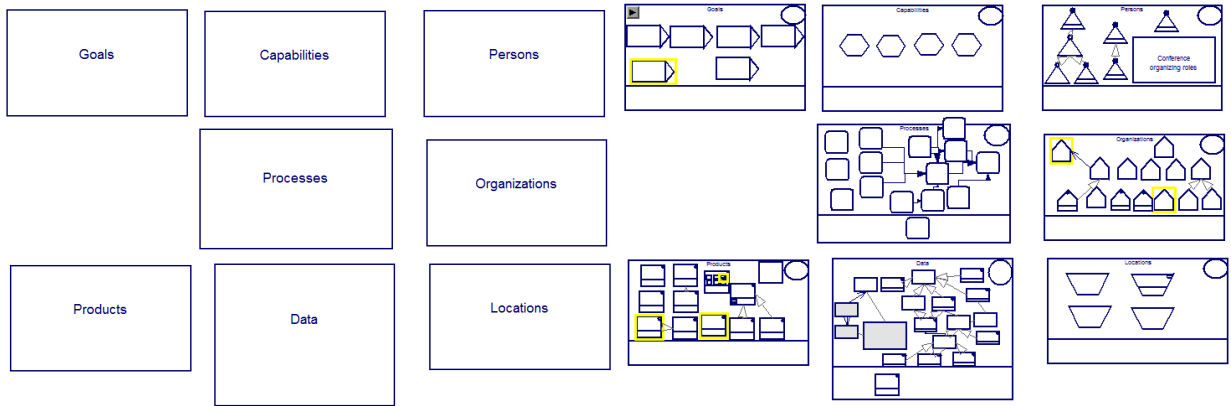
Fig. 6.8c contains a zoomed view of fig. 6.8b, where focus is on the 'Goals' thing. In 'Goals' we find a collection of GEMAL things with the 'Goal' specialisation. A similar pattern can be found in the other high-level containers for other specialisations of GEMAL thing (i.e. 'Products' for products in the bottom left part of the model, 'Persons' for persons in the top right). In addition to having a container for each of the seven specialisations of GEMAL thing, the model has one container named 'Data'. Every concept inside 'Data' is of the type GEMAL 'Thing', and a relationship structure is built among the different data types (mostly depicting a generalisation structure amongst data objects, but other relationship types than 'Generalisation' can be found there too). 'Thing' was selected as the type for data elements, since GEMAL does not have a dedicated concept for data objects.

When a user interacts with the action button in the top-left corner of the 'Goals' container, Metis transitions from the generic conference model to the related goal model (see fig. 6.8d for a fragmentation of the model). In the goal model, a structure amongst goals that are related to the conference case is found. The goal with name 'The selection process should be fair and thorough' can be seen in both models (fig. 6.8c and fig. 6.8d). This is because the goal has been copied from one of the models, and pasted as a stored mirror view in the other (denoted by the yellow border of the thing).¹

'Goal model' is merely an example of a supporting model that can be created for the 'Generic conference case' model. In a similar manner, one could create a model for process decomposition, relationships among conference participant, a model showcasing relationships between

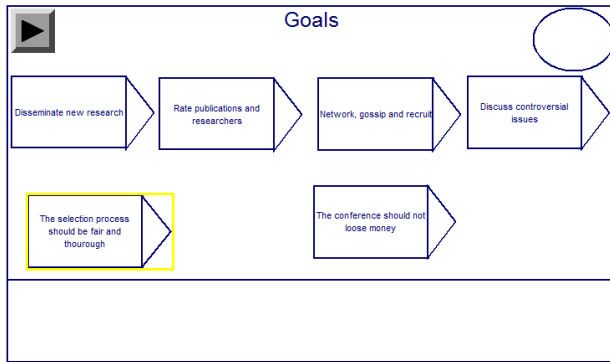
processes and products, and so on.

Inspired by this case study, a template file ('GEMAL Structured') that contains a suggested structure to be used in modeling complex, large-scale modeling tasks was created. Refer to appendix B for details.

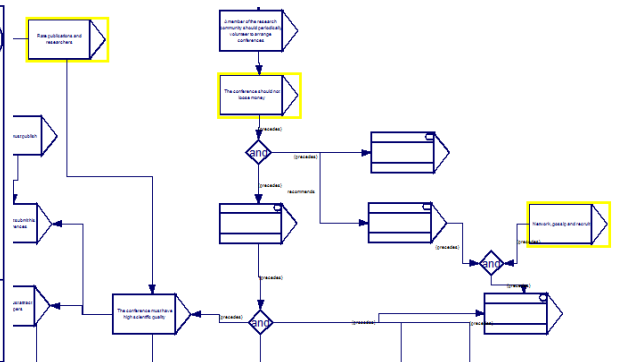


(a) A view of the generic conference model where the outmost composite things are closed

(b) A view of the generic conference model where the outmost composite things are open



(c) A sub-view of figure 6.8a, where the 'Goals' thing has been zoomed to



(d) A fragment of the goal model that is associated with the generic conference model

Figure 6.8: Figures illustrating the 'Arranging a Conference' case

¹Stored mirror view is one of several options for creating additional views of the same object in Metis, and is not functionality that has been created as a part of the GEMAL implementation. Refer to the Metis user manual for details.

Chapter 7

Final Evaluation

After GEMAL had been implemented in Metis and case studies were prepared, two different types of evaluations were carried out; an online survey to cover semiotic aspects of the language, and an expert evaluation for a broader look at the quality of the language.

7.1 Semiotic Survey

As part of the evaluation of GEMAL, an online survey was created. The survey was intended to provide some indication on the quality of the semiotics of the language, namely the symbols used for specialisations of GEMAL things.

The short survey was distributed to an arbitrary selection of friends and strangers across social media (Facebook¹) and a social networking site (Reddit²). Everyone was encouraged to answer, whether they had previous training in conceptual modeling or related topics, or not. To keep the survey's questions simple enough for untrained people to answer confidently, it is focused on the connection between symbols and concepts, to find whether novice users can identify them correctly.

This implies that the survey is shaped around Moody's principle of semantic transparency (see section 3.2.2.2). Some indication towards perceptual discriminability was collected through the comment section of the survey. The survey is not geared towards collecting data towards providing indication to the other principles of Moody, as they would require more advanced evaluation material (i.e. having novice and expert users trying out actual modeling with GEMAL).

Tables 7.1 and 7.2 provides an overview of the responses given in the survey. Refer to table E.1 in appendix E for an overview of comments shared by the survey respondents.

The survey was created using SurveyMonkey³. The survey as it appeared to survey takers

can be found in Appendix D.

7.1.1 Survey Results

	Symbol A	Symbol B	Symbol C	Symbol D	Symbol E	Symbol F	Symbol G
Capability	9.61 % (5)	50.00 % (26)	9.61 % (5)	5.77 % (3)	0.00 % (0)	17.31 % (9)	7.69 % (4)
Goal	21.15 % (11)	13.46 % (7)	15.38 % (8)	5.77 % (3)	23.08 % (12)	15.38 % (8)	5.77 % (3)
Location	11.54 % (6)	15.38 % (8)	11.54 % (6)	23.08 % (12)	1.92 % (1)	9.61 % (5)	26.92 % (14)
Person	9.61 % (5)	1.92 % (1)	44.23 % (23)	1.92 % (1)	23.08 % (12)	9.61 % (5)	9.61 % (5)
Process	38.46 % (20)	9.61 % (5)	1.92 % (1)	1.92 % (1)	15.38 % (8)	25.00 % (13)	7.69 % (4)
Product	11.54 % (6)	7.69 % (4)	0.00 % (0)	44.23 % (23)	21.15 % (11)	5.77 % (3)	9.61 % (5)
Organisation	0.00 % (0)	0.00 % (0)	15.38 % (8)	19.23 % (10)	15.38 % (8)	21.15 % (11)	28.85 % (15)

Table 7.1: Results for survey question 1: *Each of the following 7 concepts are represented by one of the 7 symbols shown above. Each symbol has a distinct shape that is meant to hint at its connection to one of the concepts. Which symbol do you think represents each concept (one selection per concept)?* The correct answer in each row is indicated by a grey background. The number of responses for each alternative is shown by the number in parentheses.

¹Facebook (<https://www.facebook.com>) is a popular online social networking service.

²Reddit (<http://www.reddit.com/>) is an entertainment, social networking service and news website where registered community members can submit content, such as text posts or direct links. Registered users can then vote submissions "up" or "down" to organize the posts and determine their position on the site's pages. Content entries are organized by areas of interest called "subreddits".

³SurveyMonkey (<https://www.surveymonkey.com>) has an online survey development platform that is cloud based ("software as a service"). SurveyMonkey provides free, customizable surveys, as well as a suite of paid back-end programs that include data analysis, sample selection, bias elimination, and data representation tools.

Alternatives	Answers
No training / experience	90.38% (47)
Moderate training / experience	3.85% (2)
Advanced training / experience	0.00% (0)
I don't know / no answer	5.77% (3)

Table 7.2: Results for survey question 2: *Have you had any training or previous experiences related to conceptual modeling or modeling languages used in business process modeling?* The number of responses for each alternative is shown by the number in parentheses.

7.1.2 Analysis of Results

Table 7.2 shows that 90.38% of the survey respondents had no training or previous experience related to conceptual modeling or modeling languages used in business process modeling. Only two respondents said that they have moderate training/experience (one of them left a comment stating that the person has 10+ years of experience in IT management, and 16+ years experience in the IT industry, ref. comment 13 in appendix E). Three respondents stated that they did not know if they had any training / had no answer to share.

The high percentage of novice users implies that the results are sound when analysing them in order to show tendencies related to Semantic Transparency (ST) of symbols in GEMAL. In section 4.2.2, we suggested that some symbols may be semantically immediate, -opaque, or -perverse. We also noted that Moody claims that how novice users infer meaning from a symbol can help point out their level of ST.

For a symbol to be semantically immediate, a novice reader should be able to infer its meaning from its appearance alone. To be able to point to a semantically immediate symbol in this survey, we could require that most respondents found the right name for it. This was not discovered for any symbols. 'Person' and 'Product' are the closest ones, with 44.23% correct answers.

Semantically opaque concepts should be expected to display an arbitrary relationship between its symbol and name. We can infer that 'Person', 'Product', 'Goal', and 'Organisation' can be classified somewhere between semantically immediate and semantically opaque symbols; the correct alternative got the most answers (or slightly lower than another alternative in the case of 'Goal') for those symbols in the survey. 'Capability', 'Location', and 'Process' may be closer to a semantically perverse classification, since there was a noticeably higher number of respondents selecting a individual wrong alternative than there were respondents selecting the correct alternative. The survey indicates that 'Capability' is the concept closest to semantically perverse; as much as 50% of the respondents thought that the right symbol was the symbol belonging to 'Location'!

Comments left by respondents at the end of the survey can provide some indication towards

the perceptual discriminability of GEMAL. Recall from section 3.2.2.2 that PD is about the ease and accuracy with which graphical symbols can be differentiated from each other. Accurate discrimination of symbols is a prerequisite for accurate discrimination of diagrams. Commenter #3 had 7 out of 9 correct answers. To mention some of his remarks for correct answers, he stated that 'Goal' looks like 'an arrow pointing towards a goal', 'Person' is 'formed like a person', and 'Product' is 'square, referring to a concrete item'. He commented that he had 'no idea' for the symbol for 'Location' and 'Capability', which were the two answers he got wrong. Commenter #12 reasoned that the symbol for 'Organisation' (alternative G) should be linked to the 'Goal' concept, because it looks like 'a box filled with the necessary tools to reach the goal that the top triangle is pointing towards'. Commenter #9, who got all the answers wrong, reasoned that 'Goal' was strong (picked the 'Person' symbol), 'Process' was moving (picked the 'Organisation' symbol), and 'Location' was firm (picked the 'Product' symbol). Four respondents stated specifically that they were confident that they got the symbol for the 'Person' concept right, and they all did. We can draw no conclusions towards the perceptual discriminability of GEMAL from these comments. However, we can say that the PD of GEMAL is far from perfect, as quite a few of the commenters provided (confident) reasoning for wrong alternative answers.

7.2 Expert Evaluation

The expert evaluation was carried out towards the end of the project, when seeking to generate indications on the quality of GEMAL as a modeling language from credible sources.

The experts were provided with the following material:

- Models in the modeling languages DFD, ER, I*, UML, and ArchiMate (the models can be seen in fig. 6.1a, 6.2a, 6.4a, 6.5a, 6.6, or in the digital appendix (appendix H)).
- GEMAL reimaginings of the same models, made with the Metis tool (the models can be seen in fig. 6.1b, 6.2b, 6.4b, 6.5b, 6.6b, or in the digital appendix (appendix H)). The models had first been approved by the original proposer of GEMAL and supervisor of this project, John Krogistie. These models were presented to the experts on the personal laptop of the student.
- A GEMAL model in Metis displaying some of the ways in which GEMAL can support conceptual and perceptual integration (can be found in the digital appendix (appendix H)).
- A printout containing an overview of symbols in GEMAL (see appendix G).

The material was sent via email to the experts a couple of days before the evaluation, along with a description of how the evaluation was intended to be carried out. The original models were brought to the evaluation as printouts, while the GEMAL models were presented on the personal laptop of the student. The student took the initiative to present the models to the

experts, but the experts were encouraged to choose models they wanted to have a closer look at, navigate themselves, or skip - depending on their interests and knowledge. Each evaluation session took about 50 minutes. The student and one expert took part in each session, and sat together in front of the laptop with the printouts readily available. Experts were encouraged to speak their mind about the models first, before the questions were answered. The student would take notes along the way.

It was strived to find experts with a diverse set of relevant specialisation areas. Two experts have completed the evaluation; one is specialised towards model-based user interface design and model-driven software development, while the other is more interested in technical information systems engineering, including topics like requirements engineering and information systems security. More detail of their background can be found in the next section. The idea here was that they would be able to shed light on GEMAL from different professional points of view. A third type of expert was contacted; someone that had expert knowledge about the Metis tool. Unfortunately, no such expert that had time to take part in an evaluation was found.

7.2.1 Results

1. About you

What previous experience do you have with conceptual modeling and related topics? Do you have experience with DFD, ER, I, UML, ArchiMate, other languages?*

Expert A: He is a researcher and educator within information systems engineering, specialising on information systems modeling, model-based user interface design, model-driven software development, design of domain-specific languages, and Eclipse application and plugin development. Fairly familiar with all the mentioned modeling languages.

Expert B: He is a researcher and educator within information systems engineering, specialising on conceptual modeling, requirements engineering, and information systems security. He is familiar with all the mentioned modeling languages, but has the most experience with DFD, ER and I*. He knows some of the UML diagram types well, and has some experience with ArchiMate.

2. Has GEMAL been successfully implemented in Metis?

With focus on the technical implementation of the language: What is done well? What could have been done better? Suggestions for improvements?

Expert A: Larger fonts and more noticable relationship lines could prove to be beneficial, it is hard to notice these things when the view is zoomed out a bit. This may

be caused by limitations in Metis. GEMAL looks a bit old fashioned in Metis, a more modern interface could be better.

Expert B: Hard to give a detailed answer here based on the evaluation walkthrough. GEMAL seems to respond fast and work well. Would have to try modeling with it himself to be able to provide suggestions for improvements.

3. Is GEMAL able to represent every relevant aspect of traditional modeling cases?

3.1. *Based on the evaluation models, which notations were represented well in GEMAL? Which were not so well represented?*

Expert A: In the UML Activity Diagram, he thinks it is a good solution to use a GEMAL 'Thing' in place of a UML 'Initial Node', this makes it possible to add additional initial properties in GEMAL models. He does not think that GEMAL needs dedicated symbols to represent fork/join in UML. In the DFD model, he view the inline information flow of GEMAL as beneficial. However, it can be difficult to separate information flow from process flow in the GEMAL diagram. For the ArchiMate model, he questioned what it means to have a process inside a product. He points out that the naming scheme of GEMAL should strive to be clear about if a word is in verb or noun form (i.e. 'my curriculum Book' or 'to book a flight'). Also, ArchiMate Datastores could be annotated with '«Datastore»' to make the GEMAL model more clear.

Expert B: Based on the evaluation models, GEMAL seemed to cover them well enough. Sequence Diagrams may be problematic to show, as may for example special constructs in Activity Diagrams. The DFD model took up less space than the GEMAL model, while details could be added to the GEMAL model without adding complexity on the surface. The GEMAL equivalent to UML Use Case looked similar to a DFD diagram, since things could be nested. The GEMAL equivalent to the ER model looks very clean, but how can details for relationships be showed clearly? i* has two types of models ('Strategic Dependency Model' and 'Strategic Rationale Model'), while only one was showed in the evaluation case. Can both be supported well in concert?

3.2. *Are there other traditional modeling cases that you can foresee will work well or not so well in GEMAL?*

Expert A: When the notation has few concepts as in GEMAL, some things cannot be expressed as easily in the models. Models leave room for interpretation, and may be understood differently by users. Better support for modeling deontic logic could be useful in some cases. He calls for a way to model the conditions of a modality property. Utilising OCL (used in UML) in GEMAL could help with this. For GEMAL relationship types, dotted lines could advantageously be given a meaning as seen in some notations; relations involved in information flow could have dotted lines,

while those involved in process flow has filled lines. Also, a hollow arrow-head could symbolise that information is sent, while a filled arrow-head means the opposite. The direction of a relation could be used to symbolise read/write operations - then perhaps 'Uses' and 'Communicates' could be combined to one relation, where the direction of the relation symbolises the meaning that they currently hold separately.

Expert B: Events in BPMN may be tricky to represent. Also specialised concepts in for example BPMN and ArchiMate. To be complete, GEMAL would need a way to represent states in state diagrams, but this may not be required in a simple language. Also, could OCL be useful to model conditions in GEMAL? OCL is used in for example UML.

- 3.3. *Does GEMAL seem to adapt better to some modeling perspectives than others? (see examples in appendix F)*

Expert A: [We did not have time to go through all the perspectives. See some related answers under question 3.2.]

Expert B: Unsure about how well behavioural languages like Petri Nets and State-charts can be modeled, GEMAL may not have the concepts required for this. Communication languages like Speech Act Models can also be a problem, hard to say based on the evaluation cases. GEMAL seems to cover many modeling perspectives with a low amount of constructs, which is good. UML has many different symbols for the 'Actor' concept, while GEMAL has one. It may require some effort to put GEMAL to use for people that are accustomed to using existing modeling languages. Case studies with modellers are needed to find out whether they would benefit from switching to GEMAL. Using GEMAL may be more tempting to new modellers than to experts.

- 3.4. *Can you see advantages or disadvantages that come with the holistic/molecular approach of the GEMAL language?*

Expert A: It provides the modeler freedom from perspectives, which can be an advantage in enterprise modeling. However, sometimes a more narrowly defined language can be easier to use, and this approach brings new user interface challenges.

Expert B: It makes it possible to model according to any perspective, which can be great, especially for high level enterprise models. May not be as useful for technical system models.

- 3.5. *Does GEMAL fit large and complicated modeling tasks?*

Expert A: [See the expert's answer to question 5.10.]

Expert B: For large tasks it is great to have the zoom and view/copy functionality that comes with Metis. It could also be useful to have ways to search for components,

follow specific connections in a model, search by criteria (i.e. 'view all goals that have not yet been decomposed'), and have tools to help get an overview of errors and incompleteness in a model.

4. **Are all notations when combining modeling concepts freely like in GEMAL meaningful?**

4.1. **Expert A:** As mentioned, concepts in GEMAL leave room for interpretation. What the writer tries to convey may not be what every user receives. As an improvement that could help alleviate misunderstandings, a 'Method' concept could be added, that would be different from 'Process'.

4.2. **Expert B:** Meaningless constructs can be created, but it is not a big problem in a language like GEMAL. modeling languages can have strict rules that may force modellers to follow a specific pattern when modeling (i.e. connection rules for an entity have to be fulfilled before more entities can be added), but in more loose languages, the modeller is given freedom to create the models how he wants. The expert thinks that this is right for GEMAL if the goal is to create a language where the modeller is free to follow any perspective and approach.

5. **Is the comprehensibility appropriateness of a molecular modeling language like GEMAL better than what can be achieved by combining traditional modeling languages?**

5.1. *Is the **number of concepts** reasonable?*

Expert A: Yes. If any more concepts are added, it can quickly become hard for users to keep aware of their separate meanings. However, a concept similar to BPMN 'state' could prove useful. GEMAL 'Goal' could perhaps be changed to a combined concept for 'Goal' and 'Condition', where the arrow of the symbol points to the left instead of right when it means 'Condition' (make the symbol point backwards to the initial conditions of a thing, and forward to the goal of the thing).

Expert B: The number of concepts in GEMAL seems reasonable. Perhaps one for 'state' could be added? The concepts needed in GEMAL would depend on what the language's intended use is.

5.2. *Is it easy to understand the **meaning of the different concepts** in GEMAL? I.e. what a person represents when modeled as a sub-concept inside a process.*

Expert A: He thinks a more illustrative symbol for 'Product' could be found. 'Person' and 'Organisation' could have been combined into 'Actor', unless the conceptualisations are given underlying properties based on their type.

Expert B: He would guess that a person inside an organisation means that he works there. A person in a location means he's at the location or relates to the location. Location in a person could also mean that this is the current location of the person. He

thinks that the meaning of sub-concepts is very open to interpretation. How to show if a person is the producer, recipient or has some other role related to a product? Sometimes the interpretation involved may cause people to understand the information contained in a diagram differently or in a wrong way. This may also depend on how much experience the person has with conceptual modeling.

- 5.3. *(Perceptual Discriminability) - Is it easy enough to differentiate concepts graphically in GEMAL? The language use mostly shapes, dual coding (with shapes and text) and colours for this purpose.*

Expert A: He had no remarks for this question.

Expert B: Yes, the components are quite distinct without being too complex. One could experiment with more colours or icons in addition to shape, but this would add complexity to the notation.

- 5.4. *(Semiotic Transparency) - Do you find the graphical representations of concept in GEMAL intuitive? I.e. the 'Person' symbol is meant to imitate a real person.*

Expert A: Perhaps a 'Pin' symbol could replace the current 'Location' symbol. The pin would be familiar to people that use application like Google Maps.

Expert B: The 'Person' symbol is intuitive, the others less so. The 'Process' symbol is familiar to people that know DFD. The 'Decision Point' symbol is a common factor in many notations, but it is not very intuitive to understand. Icons could have been used in place of figures to increase transparency, but that would make the notation more complex.

- 5.5. *(Semiotic Clarity) - Did you see examples of, or can you think of scenarios where, symbol overload (several symbols for the same concept) or symbol redundancy (the same symbol can refer to several concepts) can occur in GEMAL models?*

Expert A: No examples.

Expert B: Saw no examples of this. However, the symbols for 'Product' and 'Thing' are similar. A 'Thing' does not have to be a 'Product'?

- 5.6. *(Graphical Economy/Visual Expressiveness) - The goal is to make GEMAL easy to learn, understand and use by utilising few visual variables both at the concept level and in larger models. Shape, colour, text, connectivity and texture are the variable categories used to differentiate concepts. Has GEMAL found a good balance between use of visual variables and the degree of expressiveness of its components?*

Expert A: Hard to say, one could experiment with more visual variables, but in GEMAL's case, it seems like a good idea to keep a visually simple notation.

- 5.7. **Expert B:** Did not notice anything negative. Experiments with users would be needed to be able to say anything conclusive about this.

- 5.8. *(Complexity Management) - GEMAL uses sub-concepts (thing-in-thing) to aid complexity management. A vagueness indicator (i.e. 'Missing by purpose') is meant to help show different levels of precision in a model. Things can also be annotated with a status (planned, waiting, ready...). Does GEMAL handle complexity in a good way?*

Expert A: Few complaints about the functionality. Can the vagueness indicator be used to show shortcomings in the tool (Metis) as well as in the language (GEMAL)?

Expert B: This functionality seemed ok. Could also be good to have filter functionality, where one can choose to see only persons, only goals, or hide for example processes.

- 5.9. *Do you find GEMAL to be **flexible** in terms of what level of details and precision a model can have?*

Expert A: Yes, GEMAL seems to be more flexible than most current modeling languages. Textual differentiation is needed to model details, but this does not have to be an issue.

Expert B: Ability to model details seems good, but some concepts of notations with a large number of concepts like BPMN may be hard to model with good precision. Textual differentiation is an option here.

- 5.10. *(Cognitive Integration) - Do you think that GEMAL has good functionality for helping users understand how different models relate to each other (conceptual integration), and for navigating complex models (perceptual integration)?*

Expert A: It is good to for example be able to extract every 'Product' to a separate diagram. He would like if data generation based on a model was supported. It would be interesting to have the metamodeling functionality of Metis in a tool like CIRIUS for Eclipse; there the programmer would not be bound by settings in Metis, but be more free to customise how the tool should work. On a positive side, Metis brings a feeling of space when it zooms and animates between diagrams. It helps show how the current view relates to the model as a whole. The functionality reminds him of the 'PREZI' presentation software. It would be nice to be able to 'tag' a thing, and then extract every thing with a tag to a new view.

Expert B: I think that the navigation support from Metis is good, and it is useful to create different views with the same information. However, these views are not automatically created; the person creating the model has to build them as part of a model. Perhaps this could have been handled automatically using AI in other tools? It would be useful if the tool could suggest other parts of the model that may be related.

6. Other

- 6.1. *Do you think creating two **view styles** that can be switched between to fit different tasks and users is a good idea? Would you add or remove anything from the basic and advanced view styles?*

Expert A: Different visualisation for different users is a good idea. However, he would prefer to have the choices listed in a setting menu, where each user can choose what to display. Not a great solution to have to drag view styles into the model, but it works.

Expert B: Good idea. Different users may want different viewstyles, and they can be switched between for different tasks and purposes.

- 6.2. *Is it a good idea to show complexity, vagueness and modality visually as in GEMAL? Should other factors be displayed as well?*

Expert A: The functionality is good, but the values could perhaps have been visualised in a better way. Complexity could be indicated by folding a corner on a thing symbol, and vagueness could be shown by applying a dotted line with some texture on the border.

Expert B: This would depend on the use of a model. If risk and security is of concern, one may want to focus on those factors visually. In other situations one may want to see performance, time allocation and power consumption visually.

7.2.2 Summary and Analysis of Results

We will here summarise the feedback generated through the expert evaluation, and attempt to interpret what it can mean to current and feature versions of the GEMAL modeling language.

Toward research question # 1, the experts seem to agree that the implementation of GEMAL in Metis works in a somewhat satisfactory way. They had suggestions for improvements (including larger fonts, more noticable lines, using updated software). They also noted that it is hard to give conclusive answers towards this topic based on the limited overview they gained of the GEMAL implementation.

For research question #2, experts provided quite rich feedback. They seemed to be generally pleased with the approach and expressiveness of GEMAL, but also shared some problems they noticed. In some cases, GEMAL models seem to take up more space than the model in the original modeling language. In other cases, it is the other way around. There will be room for intepretation in GEMAL models, and advanced components of several mentioned notation may be challenging to express in GEMAL. They raised some concern towards complexity management, relationship scheme, and graphical economy. Expanding on the ideas shared for a revised relationship scheme seems like a good idea; not every visual variable in current GEMAL relationships actually carry a meaning (i.e. dotted lines). Both expert mentioned OCL as a possible addition that could benefit GEMAL, so that might be worth looking close into.

On the topic of research question #3, both experts agreed that there will be room for interpretation in GEMAL models, but they did not see this as a big concern for GEMAL as a molecular modeling language. Stricter rules may not be a good solution, because they would restrict the modeller's freedom of expression.

Toward research question #4, they seemed to agree that the number of concepts in GEMAL is reasonable; by adding more one would impair the graphical economy of the language, and thus make requirements for complexity management tougher to meet. Some changes to the current selection of concepts were suggested. It is suggested that symbols for concepts could be more illustrative. When building composites of GEMAL concepts, users may interpret the meaning of the model differently depending on their experience level and the context of use. It seems one could experiment more with visual variables for the language, but we note that the experts were positive to the current selection of visual variables overall - even if there were some interesting suggestions for possible changes. GEMAL appears to have a flexible notation, but it is dependent on textual differentiation. This can be attributed to the low number of concepts of the language. It was noted that using GEMAL in place of some large and complex notation can prove troublesome, even if no conclusive answers on this was given. Experts were positive to the current functionality for complexity management, but possible additions were mentioned. Some of the additions are likely not possible when using the Metis platform.

On the topic of GEMAL view styles, the experts seemed positive to the idea. However, they outlined possible changes to visualisation, and mentioned variables that are not currently shown visually in GEMAL that could be added, to make it easier to adapt GEMAL to some mentioned modeling scenarios. Could it perhaps be a good idea to let users define their own set of variables to be displayed visually on GEMAL concepts?

Chapter 8

Discussion

8.1 Viability of GEMAL in its current State

GEMAL currently has a stable implementation in Metis software tools. Every core concept that was part of the design for the implementation has been implemented, and is functioning as intended.

Through case studies, it has been shown that GEMAL can represent models created in several leading modeling languages that adhere to different modeling perspectives and approaches. However, the case studies produced contained relatively basic models; it would be possible to come up with modeling tasks and notations that challenge the expressiveness of GEMAL even further. Where GEMAL lacked a concept to represent a concept of another language, a textual notation could be used in its place. Note that this is fully intended behaviour (as noted in the language proposal, one may freely describe sub-concepts of the basic GEMAL things).

8.2 Recommended Design Guidelines for GEMAL models

Here we outline some recommended design guidelines and best practices that can be followed when creating GEMAL models. The material has been developed based on experiences with case studies, survey material, and expert evaluations.

- *Adhere to naming conventions for GEMAL components:* The following textual differentiation scheme is recommended for GEMAL components (this is a general guideline, and may not be practical in all cases. Users are free to adopt a different naming convention in their models; we simply recommend to be consistent about it)
 - *Capability:* built around a verb-noun combination (i.e. 'Do work', 'Write report')
 - *Goal:* built around a combination of verb, noun, and adjective (i.e. 'Not lose money', 'Good social program')

- *Location*: built around a noun (i.e. 'Store', 'My house')
 - *Organisation*: built around a noun (i.e. 'Hotel', 'Conference Organiser')
 - *Person*: built around a noun (i.e. 'John', 'Participant')
 - *Process*: built around a verb-noun combination (i.e. 'Handle payments', 'Ship goods')
 - *Product*: built around a noun (i.e. 'Shopping Cart', 'Proceedings')
 - *Property*: built around a noun (i.e. 'Product Description', 'Date Of Expiry')
- *Use textual differentiation* to visualise concepts that do not fit to any of the specialisations of GEMAL thing. For example, '«Data Store»' can be used as a label.
 - *Create new GEMAL models with the 'GEMAL Blank' or 'GEMAL Structured' templates*: This will ensure that you get the GEMAL Toolbar in the right pane of Metis, and the components that are allowed in a GEMAL model will be placed in the Metis 'Model Tree' for you.
 - *Create stored mirror views* of components that need to be displayed in more than one place. Changes to one view of the component will change the value of all its views.
 - *Use Action Buttons* to guide the model interpreter to parts of a model that you want him to see.
 - *Think about your choice of view style*: the 'GEMAL basic' and 'GEMAL advanced' view-styles may have different benefits depending on the context of use.

8.3 Limitations

There are some limitations of this study, primarily directed toward the evaluation material. Only two experts took part in the expert evaluation, which was the primary evaluation technique used to conclude the results of this project. As mentioned in a relevant section, more potential experts were contacted, and their evaluation of GEMAL was requested. Unfortunately, no other expert had time to take part in the evaluation. While the two experts provided lots of helpful and inspiring feedback, the feedback was to some extent subjective in nature (other experts with similar interests may have different opinions), and based on a brief overview of the modeling language provided across less than an hour.

Chapter 9

Conclusion and Further Work

9.1 Conclusion

This project took a language proposal for the molecular modeling language GEMAL as a starting point. The language went through an initial evaluation, before it was implemented using the Metis family of modeling language software. The implemented language has been subject to case studies comparing it to other modeling languages that follow a variation of modeling perspectives and approaches. Results from these case studies formed part of the material used in a set of expert evaluations that were performed on GEMAL towards the end of the project. In addition, an online survey focused on the use of symbols in GEMAL was conducted. Four research questions guided the work during this project:

1. Is it possible to implement a molecular modeling language such as GEMAL in Metis?
2. Is GEMAL able to represent every relevant aspect of traditional modeling cases?
3. Are all notations when combining concepts freely like in GEMAL meaningful?
4. Is the comprehensibility appropriateness of a molecular modeling language like GEMAL better than what can be achieved by combining traditional modeling languages?

Throughout this project, every concept that was carried over from the GEMAL language proposal has been implemented using Metis. In addition, several new concepts and visual variables have been introduced and implemented as work progressed. Experience has indicated that a combination of GEMAL and Metis can operate satisfactorily, but some issues and points to improve have been highlighted during evaluation.

GEMAL has proven able to represent basic models from several leading modeling languages. However, more work remains to be done in order to fully prove GEMAL as an adequate alternative to traditional modeling languages.

It has been found that meaningless constructs can be created when combining concepts freely in GEMAL. This may be an acceptable tradeoff when the modeling language can grant the

modeler freedom from modeling perspectives, enabling him or her to fully tailor the approach to be used for the task at hand.

This project has not been able to provide fully conclusive evidence as to whether the comprehensibility appropriateness of a molecular modeling language like GEMAL is better than what can be achieved by combining traditional modeling languages. An extended evaluation effort built on empirical evaluation techniques, preferably comprising modeling scenarios with potential users of the language, seem to be needed for that. However, comparing GEMAL to a renowned reference model, ontology, and scientific basis for constructing visual notations has indicated that the approach of GEMAL has several strengths, and has substantiated that it could prove worthwhile to explore the application of GEMAL to conceptual modeling cases for enterprises further.

9.2 Further Work

The expert evaluation of GEMAL helped outline several ways in which the language can be improved in future iterations, in particular in the areas of visual representations for relationships and variables to be displayed visually. Both of the consulted experts also mentioned integrating the Object Constraint Language (OCL) with GEMAL, which would be another approach that can be followed. It could prove interesting to attempt to implement GEMAL in other tools than Metis in the future, in order to more freely be able to explore ways to navigate models and customise user a user interface for the modeling language. It would be interesting to explore the viability of GEMAL as a cooperative modeling language, by having users work on models simultaneously. Metis does contain functionality for annotating objects and version control, but opportunities may be larger in other tools.

A natural next step if work is continued on GEMAL's Metis implementation would be to gather empirical evidence towards the comprehensibility appropriateness of the language. This could be done by having potential users using the language; create models, learn to use the software, estimate the meaning of complex models - and share their feedback.

Appendix A

Acronyms

AI	Artificial Intelligence	EM	Enterprise Models
ADM	Actor Dependency Modelling language	EML	Enterprise Modelling Languages
ACM	Association for Computing Machinery	EMO	Enterprise Modules
ARIS	Architecture of Integrated Information Systems	EPC	Event-driven Process Chain
ARP	Agents-Roles-Positions modelling language	ER	Entity Relationship
B2B	Business-to-business	ERL	External Rule Language
BPD	Business Process Diagram	ESOS	Enterprise Operational Systems
BPMN	Business Process Model and Notation	FG	Functional Grammar
BWW	Bunge-Wand-Weber	GE	Graphical Economy
CAiSE	International Conference on Advanced Information Systems Engineering	GEMAL	General Enterprise Modelling and Activation Language
CF	Cognitive Fit	GEMC	Generic Enterprise Modelling Concepts
CI	Cognitive Integration	GERAM	The Generic Enterprise Reference Architecture and Methodology
CM	Complexity Management	GOM	Guidelines Of Modeling
CRC	Camera-Ready-Copy	GOMS	Goals, Operators, Methods and Selection rules
CRIS	the Comperative Review of Information Systems (conference series)	GRL	Goal-oriented Requirements Language
CSCW	Computer-Supported Cooperative Work	GSM	Generic Semantic Modelling
DC	Dual Coding	ID	Identifier
DFD	Data Flow Diagram	IEEE	Institute of Electrical and Electronics Engineers
EEM	Enterprise Engineering Methodology	IFIP	International Federation for Information Processing
EEML	Extended Enterprise Modeling Language	IS	Information System
EET	Enterprise Engineering Tools	LMS	Learning Management System
		NAMA	Not Another Modeling Approach

NIAM	Natural Language Information Analysis Methodology
OMT	Object Modeling Technique
OO	Object Orientation/Object Oriented
ORM	Object-Relational Mapping
OWL	Web Ontology Language
PA	Participant Appropriateness
PD	Perceptual Discriminability
PEM	Partial Enterprise Models
PI	Perceptual Integration
QoMo	Quality of Modeling
SC	Semiotic Clarity
SeeMe	Semi-structured, Socio-technical Modeling Method
SEQUAL	Semiotic Quality Framework
ST	Semantic Transparency
STWT	Socio-technical Walkthrough Process
TOGAF	The Open Group Architecture Framework
UML	Unified Modeling Language
VE	Visual Expressiveness
YAMA	Yet Another Modeling Approach

Appendix B

Implementation Details

This appendix details part of the GEMAL implementation in Metis that were deemed too technical and detailed to be included in the main part of the report. Refer to this text for technical details on the implementation.

B.1 Metamodel Structure

Modeling languages in Metis are generally made available through metamodels. A metamodel can contain many things, i.e. types (modeling objects), typeviews and methods. The project consists of two separate parent metamodels.

- **GEMAL1.0:** Used during implementation of the GEMAL modeling language in Metis. It links to every Metis object used in the project. Every object that has been created as part of the project can be found in the metamodel. Some objects are part of the Metis standard package; these are not directly stored with the metamodel. This metamodel can be accessed by metamodelers that seek to alter parts of the GEMAL language, or add new modeling elements to it. Advanced Metis users may find it useful to view this metamodel to learn more about how the GEMAL language functions.
- **GEMAL:** Contains a selection of elements from the 'GEMAL1.0' metamodel. As this is the metamodel that will be used by modellers, it contains a subset of modeling items that are needed for modeling. Items like typeviews, methods, criteria and primitive types are not included here.

B.2 Elements in a Metamodel

Here an overview of the different elements that can be contained in a metamodel is given. Examples of GEMALs use of these elements is given where applicable.

- **Type:** This element includes 'Object Type', 'Relationship Type', 'Abstract Type' and 'Interface Type'. In GEMAL 'Person' is a 'Object Type', 'Instance of' a 'Relationship Type' and 'Gemal abstract relationship' a 'Abstract Type'. Different sets of attributes are available to each subtype. Types are generally used to create objects and relationships to be made available through a metamodel. Abstract types and interface types are used for structuring purposes.
- **Primitive Type:** These are data types that have no parts, such as integers or strings. GEMAL uses integer primitive types to create value ranges for attributes. An example is 'Status Color Mapping' that holds the range of possible status colours for (specialisations of) thing, and their corresponding names. 'Instantiation Level' and 'Modality' are other examples of primitive types in GEMAL that have similar purposes.
- **Type View:** Specifies how to represent objects in a model view, for example, what symbol to use to represent an object view in both an open and closed state, whether to use tree or nested structures when displaying decomposable objects, and what icon to use to represent the object type in the tree structures in the left pane of the Metis window. In GEMAL, every specialisation of thing uses nested structuring for decomposition, and symbols for objects in open state have generally been made as simple as possible, to make room for internal decomposition structures.
- **View Style:** A view style can contain a way of representing objects and relationships on screen. View styles are sometimes used for choosing representations that are more intuitive to a particular audience. Methods are used to switch between each set of view styles. GEMAL has two view styles, 'basic' and 'advanced', that are useful in different scenarios and to different audiences. More about this somewhere else [TODO:ref]
- **Symbol Palette:** Symbol palettes are used to categorise and group related symbols. Sub-metamodels can have their own symbol palette, and palettes can be arranged in hierarchies. The symbol palette is automatically loaded each time a related model file is opened. Every GEMAL symbol can be found in a symbol palette.
- **Symbol:** Symbols are used to graphically represent objects and relationships in a model view. Graphics for symbols can be imported in several formats, including .jpg, .bmp, .png, .xbm and .wmf. GEMAL objects have separate symbols for 'open' and 'closed' state. Most objects also have a separate closed symbol to be used with the advanced GEMAL viewstyle.
- **Method:** Methods contain actions initiated by the user that is performed on a model. Methods can automatically arrange a set of objects, import a file, show an online document and much more. Metis includes a number of built-in methods. In GEMAL methods

are used to compute values for colours and complexity representation, and for opening documents.

- **Criteria:** A criteria specifies something to search for in a model view, and can be combined to make more complex searches. When searching in a model view, each object is evaluated against the criteria. In addition to some standard search criteria, GEMAL uses a criteria when calculating the complexity (number of sub-elements) of objects.

B.3 Files in the GEMAL Metamodel

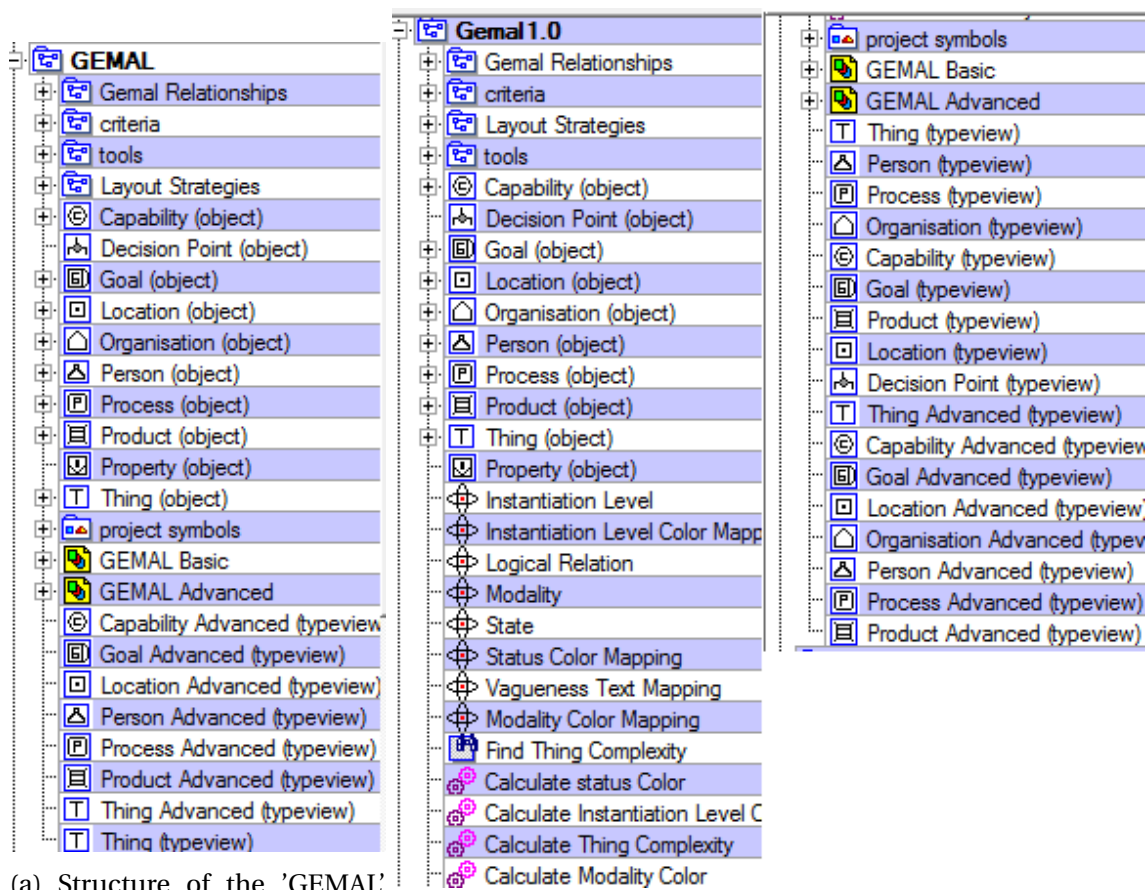


Figure B.1: Structure of GEMAL metamodels, as they appear in Metis

Fig. B.1 shows how the structure of each of the two available GEMAL metamodels appear in Metis. The files with a '+' sign can be expanded to reveal more files. Parts of the metamodels are distinguished by name, element type annotation (i.e. object, typeview), and icon. Refer to section B.2 for an explanation of element types in the GEMAL implementation.

B.4 Template Files

The GEMAL implementation has two available template files, from which users can select one when intending to create a new GEMAL model. The available template files are:

- *GEMAL Blank*: The simplest of the two. Has a toolbar containing the GEMAL concepts, and the content of the 'GEMAL' metamodel is made available in the model tree
- *GEMAL Structured*: In most ways similar to GEMAL Blank, but this template suggests a structure to be followed in large and complicated modeling tasks where GEMAL is used

An overview of the GEMAL Structured template can be seen in fig. B.2. Inspired by the findings of case study #6 (section 6.6), a container has been created for every specialisation of thing, plus a container for 'Data'. In the bottom, tabs that lead to models that the template suggests to use for modeling an individual type of thing specialisation can be found. Note that the containers of fig. B.2 have been given a name on the form 'Specialisation (Room type)' (i.e. 'Person (Bedroom)' or 'Data(Basement)'). The room type has been annotated in parantheses in an effort of helping users build a complete picture of the model; i.e. a 'Person' typically has a personal/shared bedroom, most 'Processes' in a typical home may happen in the living room.

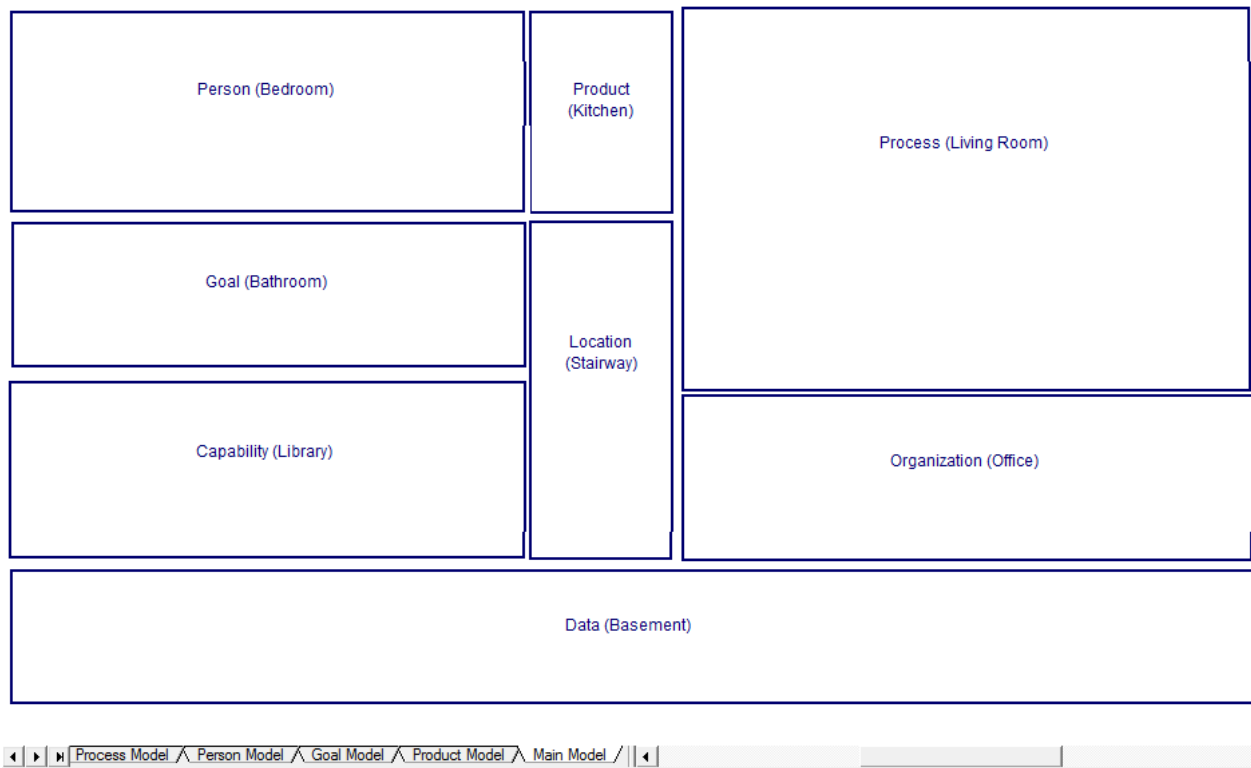


Figure B.2: Parts of the 'Main Model' default model in the 'GEMAL Structured' template

Appendix C

Case Study Descriptions

The case study descriptions that follow in section [C.1](#) and [C.2](#) have inspired most modeling tasks that have been carried out using GEMAL in this project. Case [C.2](#) was originally written by John Krogstie, and published in his book 'Model-based development and evolution of information systems: A Quality Approach' [45]. It is reproduced here with the approval of the author, to make referencing it easier for readers of this report. Case [C.1](#) has been custom written for this project.

C.1 Order Management System

A Norwegian company runs an online book shop in which customers can order books that get delivered to their home address. The shop is built around a web system. A customer seeking to order books must first register to a user account, or create a new user account in the web system. When new user accounts are created, a password, name, and creation date are stored, along with the username, address, phone number, and email of the customer. The system keeps track of the last time a customer signed in to an account.

A user account can be shared by several customers. The time an individual customer last accessed one of his associated accounts is stored along with the time the customer first gained access to the account. The system has an authentication service, that will authorise registered users that log in. The system also has an identity provider service, that keeps track of user interactions within the system.

When logged in to a user account, the customer may browse books, add books to an order, and check out when his order is complete. Orders can contain any quantity of books. A set of metadata is stored for each order, along with an unique order ID. The customer can add or remove items from his order before checking out, by interacting with the shopping cart in the web system. The system will receive confirmed orders, and start fetching the requested books. Parallel to this, the customer is requested to pay the order via invoice or credit card. The customer can

either be satisfied with the regular delivery service, or pay extra for expedited overnight delivery. Payment must have reached the book company before the order is considered terminated. Customers can achieve discounts based on how frequently they order books through the company's preferred customer program.

For books, the current quantity in stock is stored along with product ID, price, and a name. The system also stores information about individual book instances; their unique ID is stored along with the condition of that particular book. Books are ordered by the firm from manufacturers. For every affiliated manufacturer of books, an ID, name, and location is registered in the web system.

The system contains an inventory system component that connects to a inventory database, where information about books in stock is stored. A customer database contains information about registered customers, and a transaction database keep track of transactions that go through the system. Payments are treated through the store's ERP component. The firm is co-operating with a third-party firm that handles credit card information.

C.2 Arrangement of International Conferences and Events

C.2.1 Overall Setting (Basis for Type-Level Model)

An international group is involved in running a conference series, trying to take into account all the needs, both of those arranging a conference in the series and those participating in the conference in the series. In addition to supporting both conference organisers and conference participants, it aims to support the integration with the systems of travel service providers, conference locations, publishers of conference proceedings, tourist boards and professional societies (e.g. IEEE, ACM and IFIP). Organisations such as universities and research organisations arrange irregularly international scientific conferences within a specific theme. The conference is usually part of an annual or bi-annual series of similar conferences. A conference can consist of presentation of accepted articles and invited presentations, poster sessions and demonstrations/stands, panels, workshops and tutorials. Accepted articles shall be available to the conference participants at the start of the conference in the form of a so-called conference proceedings developed by a professional publisher. Who will be the organiser of the conference in a series in a particular year is decided by the conference board at annual meetings, based on applications from interested research groups. In connection to a particular conference, an organisation committee and a program committee is established. The program committee consists of a number of researchers working within the theme of the conference that are normally distributed across the world. To get good papers, the program committee of the conference announces a Call for Papers. Potential authors receive this, and some of them decide to send in one or more paper

for review. The paper can be written by one or more persons. The paper must be sent to the program chair (the leader of the program committee) within an announced deadline. The article cannot at the same time be submitted to a new event. One of the authors functions as a contact person to the program chair. When the article is received by the program chair, the contact person gets a receipt, and the article is sent to between three and five persons of the program committee for review. The reviews have to be returned by a certain date. Based on the returned reviews, a number of the articles are selected on a program committee meeting, which can be a physical or virtual meeting. Papers that have fewer than three reviews or very varying reviews will be given an additional review during the meeting. Papers can be accepted either as full papers, short papers, posters or be rejected. Accepted papers are bundled into sessions in the conference program, usually consisting of between two and four papers within the same sub-theme. After the decision on selection has been made, this is announced to the contact persons of each paper. Authors of accepted papers (full or short) must create a final version of their article, a so-called camera-ready copy (CRC), within a predefined deadline. The CRC must follow the size limitations and layout rules of the publisher of the proceedings. In addition to this, the authors are expected to take into account the comments by the reviewers when they are preparing the CRC. For certain papers one might use so-called shepherds to follow-up that requested changes are actually performed or there is a proper argumentation if this is not the case. At least one of the authors must register for the conference and present the paper in the allotted slot in the conference program. Up until the point of acceptance of the article, the author is free to withdraw the paper, but this is not possible after the final version is submitted, and the copyright transfer form is sent to the publisher of the proceedings. Note that when you submit a paper in the first place, there is already pending obligations on you (or one of the co-authors) to go to the conference and present the paper if it is accepted (note also that presenters at such conferences normally pay the registration fee). Those with invited presentations, tutorials, panels or posters must also send in a description of their presentation to the program chair within the CRC deadline, but their work does not undergo the same thorough review process. Within research, you can find different research communities, i.e. collection of researchers interested in a certain research area or research theme. In general, it is obligatory for a research community to spread scientific results, either through arranging scientific conferences or by publishing journals. One thing that characterises a scientific conference is that the papers to be presented are selected through a detailed review process as described above. The same community would thus need to arrange conferences within their research field. It is quite a lot of work arranging conferences of this type. Thus it is normal that a conference series moves between different locations each year, being arranged by one of the organisations that are part of the community. Since there is a certain economic risk in arranging a conference, it is important that the conference is successful in the sense that it attracts a sufficient number of paying participants. One

way to achieve this is to attract a number of good papers to the conference and to create a good program. Parts of a good program might be panels, tutorials, workshops, poster/demo sessions and invited presenters (so-called keynote speakers that are famous within their research field). In addition, the social program of the conference might be important in this respect given the opening for building a common history of the research community. An organisation arranging a scientific conference depends on a number of other people and organisations for the practical arrangements of the conference. Whereas the organising and program committees are typically recruited from the research community and do the work relating to securing the quality of the scientific program for free, there need to be facilities for holding the conference, housing the participants, supporting their trip to the conference, taking care of payment and money matters, publishing information on the web, arranging social events part of the social program, performing registration at the start of the conference and publish proceedings. Many of these tasks can be supported by a conference arrangement organisation, whereas other are supported by organisations such as travel arrangers, local tourist offices, conference venues, hotels and publishers (for the proceedings). All of these services cost money, which have to be balanced by the income mainly through the participant fee and from sponsors. Thus, a budget needs to be developed to guide the choices of the organising committee. In connection to a scientific conference, it is as stated above important that the best papers are accepted. The authors are those with primary interest in getting the papers published, and should not be able to influence the review process in a subjective way. On the other hand, people in the program committee are usually allowed to submit papers to a conference. For smaller conferences/workshops, even the organising committee might be allowed to submit papers. To improve the objectivity, it is normal that the review process is so-called double blind, i.e. that you do not know who has written the paper you review, and the authors do not know who has performed the review. The overall review process is normally managed through a web-based paper management system. This system can be configured to follow a number of variants of the review process described above. To make registration and payment easy, it is usually possible to do this using credit card over the Internet. This also applies for booking hotel rooms and for travel arrangements in general.

C.2.2 Basis for Instance-Level Model (Most Information Found on the Web)

One of the first scientific conferences of this type to be supported by the system was the CAiSE conference series. CAiSE'07 was held in Trondheim 11-15 June 2007 (see <http://caise07.idi.ntnu.no>) CAiSE'08 was held in Montpellier, France. CAiSE'07 was arranged at Britannia Hotel (<http://www.britannia.no>) with the support of NTNU Videre (<http://www.ntnu.no/videre/konferanser.html>) for the practical arrangements.

Appendix D

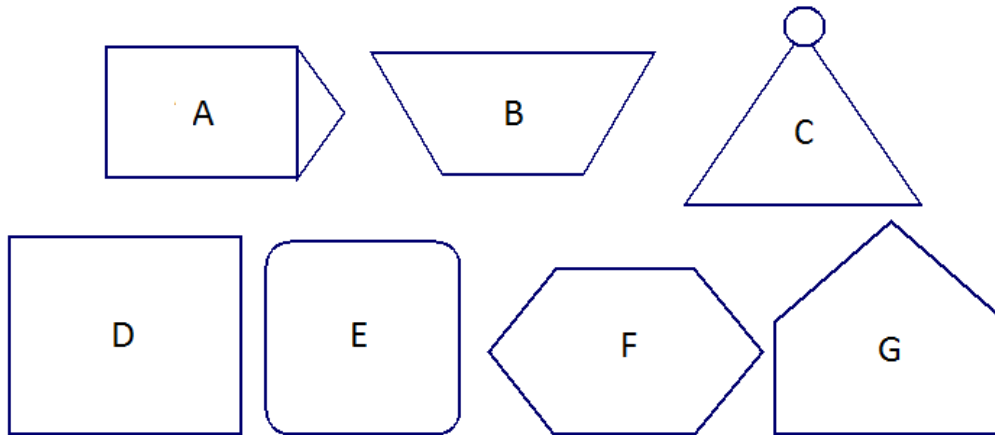
Online Survey

This is a short survey that has been developed as part of a master's thesis in Computer Science at the Norwegian University of Science and Technology (NTNU) in Trondheim, Norway. The intent is to find out whether the connection between a set of symbols and a set of concepts used in a modelling language (GEMAL) is intuitively understandable to potential users.

Your answers are valuable both if you have little or no training in conceptual modelling, or if you consider yourself an expert on the topic. The results of the survey will only be used in material related to the master's thesis.

You can expect that the survey takes a couple of minutes to complete. Thank you for your contribution!

Fig 1: The following are 7 symbols used in the conceptual modelling language GEMAL:



*** 1. Each of the following 7 concepts are represented by one of the 7 symbols shown above. Each symbol has a distinct shape that is meant to hint at its connection to one of the concepts. Which symbol do you think represents each concept (one selection per concept)?**

	Symbol A	Symbol B	Symbol C	Symbol D	Symbol E	Symbol F	Symbol G
1. Capability	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
2. Goal	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
3. Location	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
4. Person	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
5. Process	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
6. Product	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
7. Organisation	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

*** 2. Have you had any training or previous experiences related to conceptual modelling or modelling languages used in business process modelling?**

- No training / experience
- Moderate training / experience
- Advanced training / experience
- I don't know / no answer

comment (optional)

You are encouraged to comment on whether you think that you found a connection between some of the symbols and concepts mentioned in question 1. Other comments are also appreciated.

3. Comments (optional):

Ferdig

[Drevet av SurveyMonkey](#)
[Opprett din egen gratis nettbaserte spørreundersøkelse nå!](#)

Appendix E

Survey Comments

Table E.1: Comments stated by respondents of the GEMAL online survey

ID	Comment	Correct Answers	Time Spent (mm:ss)
1	Person and goal I think was right. G I think is location or organisation.	100%	09:07
2	Im clueless on this topic im afraid	14.29%	03:50
3	How I came up with the answers: A: Seems like an "arrow" pointing towards a goal B: No idea C: Formed like a person D: The shape is square, referring to a concrete item, i.e. a product F: No idea E: Rounded, less concrete than a product, thereby a process G: Seems like the structure of an organisation (many people at the bottom, few at the top)	71.43%	15:46
4	I have a feeling that my answers are correct because I think the figures look alike my answers.	42.86%	06:39
5	<i>[Translated from Norwegian]</i> I was unsure about most of them, but pretty sure C was correct.	00.00%	10:11
6	<i>[Translated from Norwegian]</i> I hope some of them were right. Jarand:)	28.57%	06:37
7	<i>[Translated from Norwegian]</i> Unsure about them all. Person and location may have been correct.	28.57	03:52

Continued on next page

Table E.1 – continued from previous page

ID	Comment	Correct Answers	Time Spent (mm:ss)
8	I think the first three are correct, but unsure about the others.	42.86%	04:50
9	Goal was strong - process moving - location firm.	00.00%	16:51
10	<i>[Translated from Norwegian]</i> This was all Greek to an old aunt. Good luck??	00.00%	09:58
11	I thought it was hard to find a connection spot on, and Im not sure if it would be the same result for me if I did the test again some other day.	00.00%	06:41
12	Capability – I chose (F) as the shape for me signals "inclusive", a bit like someone with open arms showing "come here, I can help". Process – I chose (E) as I think of the shape with rounded corners as a "busy cursor" when your computer is busy running a process. My mind almost automatically set some sort of clockwise motion around the edge. Goal – The housey shape (G) is in this case interpreted as a box filled with the necessary tools to reach the goal that the top triangle is pointing towards.	28.57%	12:18
13	10+ years of management 16+ years of IT industry experience	57.15%	04:56
14	I hope I got one right:)	28.57%	05:04
15	this was a real struggle i hope that my answer helped you	42.86%	03:23
16	Hard to come up with answers with so little context. I think C is correct because it looks like a person. D and E I think are right but they may be switched up. G organisation or maybe location	57.14%	08:23
17	<i>[Translated from Norwegian]</i> I did not understand all the words, but tried my best!!	28.57%	04:13
18	My reasons: A is goal, because it looks like a football goal in 3D. C is person, because it has a face. F is location, because it looks like a bullseye. G is organisation, because it looks like a house. B D E I had to guess.	57.14%	08:28

Continued on next page

Table E.1 – continued from previous page

ID	Comment	Correct Answers	Time Spent (mm:ss)
19	A: B: goal (as a jigsaw) C: person with head and body D: flat as a produkt box E: process is a round produkt F: capability? G: organisation (as a house)	71.43%	11:31

Appendix F

Expert Evaluation Questions

GEMAL Expert Evaluation Questions

1. About you

- 1.1. Do you have any previous experience with conceptual modeling and/or related topics?
- 1.2. Do you have experience with DFD, ER, I*, UML, ArchiMate, BPMN? Other languages?

2. Has GEMAL been successfully implemented in Metis?

With focus on the technical implementation of the language:

- 2.1. What is done well?
- 2.2. What could have been done better?
- 2.3. Suggestions for improvements?

3. Is GEMAL able to represent every relevant aspect of traditional modelling cases?

- 3.1. Based on the evaluation models, which were represented well in GEMAL? Which were not so well represented?
 - a) DFD
 - b) ER
 - c) I*
 - d) UML (Class Diagram, Use Case, Activity Diagram)
 - e) ArchiMate
- 3.2. Are there other traditional modelling cases that you can foresee will work well or not so well in GEMAL?
- 3.3. Does GEMAL seem to adapt better to some modelling perspectives than others?
Examples:
 - a) **Behavioural** – Petri Nets, Statecharts..
 - b) **Functional** – DFD, BPMN, UML Activity Diagrams
 - c) **Structural** - ER-diagrams, ORM, WordNet (AI)
 - d) **Goal and Rule** – OWL, ERT, ERL
 - e) **Object Oriented** – OMT, UML
 - f) **Communication** – Action Workflow, Speech Act Models
 - g) **Actor and Role** – I*, e³
 - h) **Topological** – place oriented, floor plan visualisation, temporal awareness
 - i) **Other perspectives** - data, process, event/behaviour, role, organisational, informational
- 3.4. Can you see advantages or disadvantages that come with the holistic/molecular approach of the GEMAL language?
- 3.5. Does GEMAL fit large and complicated modeling tasks?

4. Are all notations when combining modelling concepts freely like in GEMAL meaningful?

- 4.1. Can you think of **meaningless constructs** that can be created in GEMAL?
- 4.2. Do you think this is problematic?
- 4.3. Any suggestions for how this issue could be improved?

5. Is the comprehensibility appropriateness of a molecular modelling language like GEMAL better than what can be achieved by combining traditional modeling languages?

In the following questions, please comment on things that are done well, not so well, and state suggestions for improvements or changes if any, based on the evaluation models you have seen:

- 5.1. Is the **number of concepts** reasonable? Are they appropriately specialised/generalised?
- 5.2. Is it easy to understand the **meaning of the different concepts** in GEMAL? I.e. what a person represents when modeled as a sub-concept inside a process.
- 5.3. **Perceptual Discriminability** – Is it easy enough to differentiate concepts graphically in GEMAL? The language use mostly shapes, dual coding (with shape and text) and colors for this purpose.
- 5.4. **Semiotic Transparency** – Do you find the graphical representations of concepts in GEMAL intuitive? I.e. the 'Person' symbol is meant to imitate a real person.
- 5.5. **Semiotic Clarity** – Did you see examples of, or can you think of scenarios where, symbol overload (several symbols for the same concept) or symbol redundancy (the same symbol can refer to several concepts) can occur in GEMAL models?
- 5.6. **Graphic Economy/Visual Expressiveness** – The goal is to make GEMAL easy to learn, understand and use by utilising few visual variables both at the concept level and in larger models. Shape, color, text, connectivity and texture are the variable categories used to differentiate concepts. Has GEMAL found a good balance between use of visual variables and the degree of expressiveness of its components?
- 5.7. **Complexity Management** – GEMAL uses sub-concepts (thing-in-thing) to aid complexity management. A vagueness indicator (i.e. 'Missing by purpose') is meant to help show different levels of precision in a model. Things can also be annotated with a status (planned, waiting, ready...). Does GEMAL handle complexity in a good way?
- 5.8. Do you find GEMAL to be **flexible** in terms of what level of detail and precision a model can have?

5.9. **Cognitive Integration** – Do you think that GEMAL has good functionality for helping users understand how different models relate to each other (conceptual integration), and for navigating complex models (perceptual integration)?

a) **Conceptual Integration** - GEMAL has mechanisms from Metis tools aimed at helping a model interpreter assemble information from separate diagrams into a coherent mental representation of a system. Top-level models can be created for more specialised sub-models (long-shot diagram).

b) **Perceptual Integration** – Buttons can be used to transition between diagrams. This functionality aims to help a model interpreter understand where he is, where he can go, if he's on the right way, and understand where the model ends.

6. Other

6.1. Do you think creating two **view styles** that can be switched between to fit different tasks and users is a good idea? Would you add you remove anything from the basic and advanced GEMAL view styles?

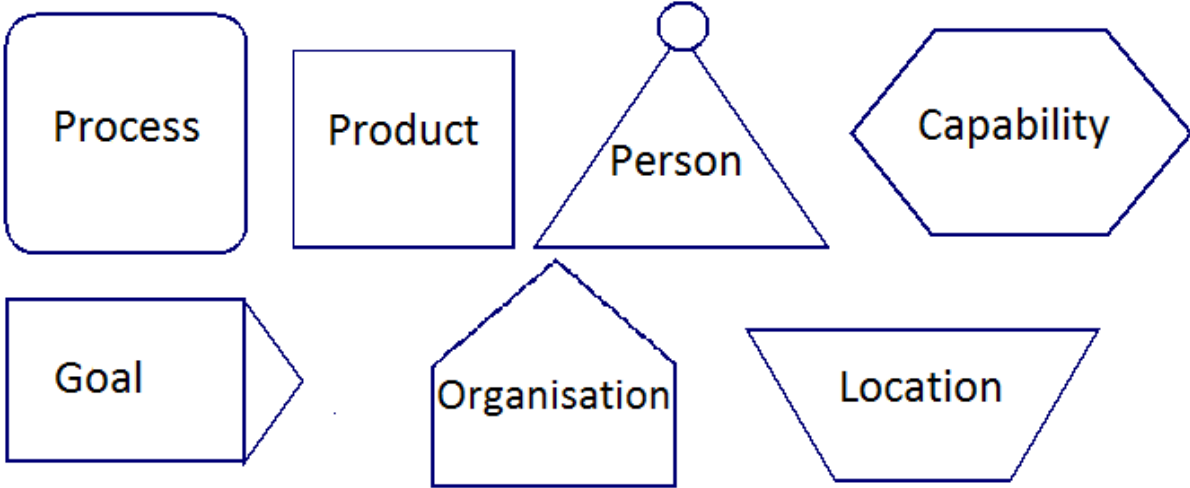
6.2. Is it a good idea to show complexity, vagueness and modality visually as in GEMAL? Should other factors be displayed visually as well?

Appendix G

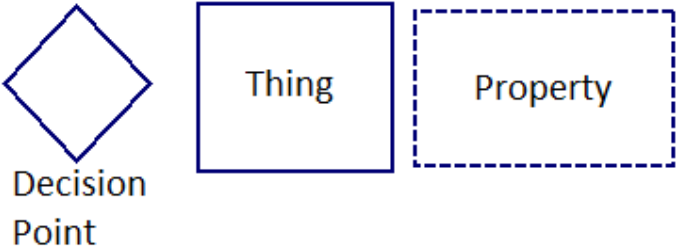
GEMAL Language Overview

GEMAL Language Overview

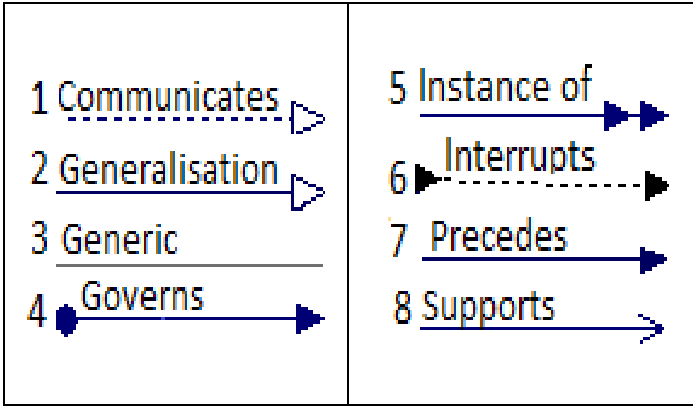
Specialisations of 'Thing':



Other Symbols:



Relationship Symbols:



Advanced View Style:

The diagram illustrates an 'Advanced View Style' for a product. It features a large gray rectangular area with a dark blue border. In the top right corner of this area is a green circle containing the word 'Ready'. Centered in the gray area is the text 'My Product'. At the bottom of the gray area is a horizontal bar divided into three segments: a white segment on the left containing the number '1', a green segment in the middle containing the word 'Complete', and a yellow segment on the right containing the word 'obligation'.

Appendix H

Digital Attachments

Here follows a list of digital attachments, with a short description for each part:

1. **Evaluation_material**

- (a) **Evaluation_cases_original**: Contains figures used for models in other modeling languages than GEMAL in .png format:
 - i. ArchiMate_order_case
 - ii. DFD_order_case
 - iii. ER_order_case
 - iv. ISTAR_conference_case
 - v. UML_order_activity_diagram
 - vi. UML_order_class_diagram
 - vii. UML_order_use_case_diagram
- (b) **GEMAL_evaluation_cases**: Contains GEMAL versions of the figures mentioned above. Also contains some other GEMAL models used during expert evaluation

2. **GEMAL_implementation**: Contains files required for the Metis implementation of GEMAL

- (a) **http**: Contains all files that are part of the implementation of the GEMAL metamodels. Should be copied to METISHOME/xml/http
- (b) **startup**: Contains references to startup files and template files for the Metis implementation. Should be copied to METISHOME/xml/startup

3. **GEMAL_models**: Contains a selection of GEMAL models for Metis that have been developed as a part of this project. The models have been referred to from parts of this report

- (a) **conforg**: Created by John Krogstie. Related to section 6.6 about case study #6.
- (b) **gemal-metamodel**: Metis implementations of the figures referred to in in fig. 5.1 and fig. 5.2

Bibliography

- [1] Archimate ®| the open group. <http://www.opengroup.org/subjectareas/enterprise/archimate>. Accessed: 2015-01-11.
- [2] Archimate ®2.1 specification. http://pubs.opengroup.org/architecture/archimate2-doc/chap02.html#_Toc371945149. Accessed: 2015-01-11.
- [3] Heuristic evaluations and expert reviews | usability.gov. <http://www.usability.gov/how-to-and-tools/methods/heuristic-evaluation.html>. Accessed: 2015-01-13.
- [4] Togaf ®version 9.1. <http://www.opengroup.org/togaf/>. Accessed: 2015-01-11.
- [5] Uml [omg web site for uml specifications]. <http://www.omg.org/spec/UML/>. Accessed: 2015-01-13.
- [6] Robert L Ashenurst. Ontological aspects of information modeling. *Minds and Machines*, 6(3):287–394, 1996.
- [7] Esa Auramäki, Erkki Lehtinen, and Kalle Lyytinen. A speech-act-based office modeling approach. *ACM Trans. Inf. Syst.*, 6(2):126–152, April 1988.
- [8] D.E Avison and G. Fritzegerald. *Information systems development: methodologies, techniques and tools*. McGraw-Hill, London, 1995.
- [9] Peter Bernus and Laszlo Nemes. A framework to define a generic enterprise reference architecture and methodology. *Computer Integrated Manufacturing Systems*, 9(3):179–191, 1996.
- [10] Jacques Bertin. *Semiology of Graphics: Diagrams, Networks, Maps*. University of Wisconsin Press, 1983.
- [11] N. Block. Holism, mental and semantic. *Routledge Encyclopedia of Philosophy*. Routledge, New York, Ny, USA, 1998.
- [12] J. Bubenko, C. Rolland, P. Loucopoulos, and V. DeAntonellis. Facilitating “fuzzy to formal” requirements modelling. In *Requirements Engineering, 1994., Proceedings of the First International Conference on*, pages 154–157, Apr 1994.
- [13] Mario Bunge. *Treatise on basic philosophy: Ontology I: the furniture of the world*, volume 1. Springer, 1977.
- [14] Johannes Franciscus Maria Burg. *Linguistic instruments in requirements engineering*. IOS Press Amsterdam, 1997.
- [15] Edward S Casey. *The fate of place: A philosophical history*. Univ of California Press, Berkeley, 1998.
- [16] Peter Pin-Shan Chen. The entity-relationship model—toward a unified view of data. *ACM Transactions on Database Systems (TODS)*, 1(1):9–36, 1976.
- [17] Bill Curtis, Marc I. Kellner, and Jim Over. Process modeling. *Commun. ACM*, 35(9):75–90, September 1992.
- [18] G. B. Davis. Strategies for information requirements determination. *IBM Syst. J.*, 21(1):4–30, March 1982.
- [19] Tom DeMarco. *Structured analysis and system specification*. Yourdon Press, 1979.

- [20] Paul Dourish. Using metalevel techniques in a flexible toolkit for cscw applications. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 5(2):109–155, 1998.
- [21] Paul Dourish. Re-space-ing place: place and space ten years on. In *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, pages 299–308. ACM, 2006.
- [22] Eckhard D Falkenberg. *A Framework of Information System Concepts: The FRISCO Report (Web edition)*. University of Leiden, Department of Computer Science, 1998.
- [23] Tim Finin, Richard Fritzson, Don McKay, and Robin McEntire. Kqml as an agent communication language. In *Proceedings of the third international conference on Information and knowledge management*, pages 456–463. ACM, 1994.
- [24] ACL Fipa. Fipa acl message structure specification. *Foundation for Intelligent Physical Agents*, <http://www.fipa.org/specs/fipa00061/SC00061G.html> (30.6. 2004), 2002.
- [25] C. Floyd. A comparative evaluation of system development methods. In *Proc. Of the IFIP WG 8.1 Working Conference on Information Systems Design Methodologies: Improving the Practice*, pages 19–54, Amsterdam, The Netherlands, The Netherlands, 1986. North-Holland Publishing Co.
- [26] Andrew Gemino and Yair Wand. Empirical comparison of object-oriented and dataflow models. In *Proceedings of the Eighteenth International Conference on Information Systems, ICIS '97*, pages 446–447, Atlanta, GA, USA, 1997. Association for Information Systems.
- [27] Sundar Gopalakrishnan, John Krogstie, and Guttorm Sindre. Adapting uml activity diagrams for mobile work process modelling: Experimental comparison of two notation alternatives. In *The Practice of Enterprise Modeling*, pages 145–161. Springer, 2010.
- [28] Sundar Gopalakrishnan, John Krogstie, and Guttorm Sindre. Extending use and misuse cases to capture mobile applications. *NOKOBIT*, 2011, 2011.
- [29] J. Gordijn, E. Yu, and B. van der Raadt. E-service design using i^* and e^3 value modeling. *Software, IEEE*, 23(3):26–33, May 2006.
- [30] David G Green and David Newth. Towards a theory of everything?—grand challenges in complexity and informatics. *Complexity international*, 8(5):1–12, 2001.
- [31] Terry Halpin and Tony Morgan. *Information modeling and relational databases*. Morgan Kaufmann, 2010.
- [32] Steve Harrison and Paul Dourish. Re-place-ing space: the roles of place and space in collaborative systems. In *Proceedings of the 1996 ACM conference on Computer supported cooperative work*, pages 67–76. ACM, 1996.
- [33] Jeff Heflin. Owl web ontology language-use cases and requirements. *W3C Recommendation*, 10:12, 2004.
- [34] Thomas Herrmann et al. Seeme in a nutshell. the semi-structured socio-technical modeling method. *Ruhr-Universität Bochum, Bochum*, 2006.
- [35] Thomas Herrmann, Marcel Hoffmann, Kai-Uwe Loser, and Klaus Moysich. Semistructured models are surprisingly useful for user-centered design. *Designing Cooperative Systems (Coop 2000)*, pages 159–174, 2000.
- [36] Thomas Herrmann and Kai-Uwe Loser. Vagueness in models of socio-technical systems. *Behaviour & Information Technology*, 18(5):313–323, 1999.
- [37] Thomas Herrmann, Kai-Uwe Loser, and Klaus Moysich. Intertwining training and participatory design for the development of groupware applications. In *PDC*, pages 106–115, 2000.
- [38] Juhani Iivari and Pentti Kerola. A sociocybernetic framework for the feature analysis of information systems design methodologies. *Information systems design methodologies: A*

- feature analysis*, pages 87–139, 1983.
- [39] Henry Jackman. Moderate holism and the instability thesis. *American Philosophical Quarterly*, pages 361–369, 1999.
- [40] M. Jarke, J. Bubenko, C. Rolland, A. Sutcliffe, and Y. Vassilou. Theories underlying requirements engineering: an overview of nature at genesis. In *Requirements Engineering, 1993., Proceedings of IEEE International Symposium on*, pages 19–31, Jan 1993.
- [41] Håvard D. Jørgensen. *Interactive Process Models*. PhD thesis, Norwegian University of Science and Technology, Faculty of Information Technology, Mathematics and Electrical Engineering, 2004.
- [42] Steven Kelly, Kalle Lyytinen, and Matti Rossi. Metaedit+ a fully configurable multi-user and multi-tool case and came environment. In Panos Constantopoulos, John Mylopoulos, and Yannis Vassiliou, editors, *Advanced Information Systems Engineering*, volume 1080 of *Lecture Notes in Computer Science*, pages 1–21. Springer Berlin Heidelberg, 1996.
- [43] Gregor Kiczales. Beyond the black box: Open implementation. *Software, IEEE*, 13(1):8–10, 1996.
- [44] John Krogstie. A semiotic approach to quality in requirements specifications. In *Organizational Semiotics*, pages 231–249. Springer, 2002.
- [45] John Krogstie. *Model-based development and evolution of information systems: A Quality Approach*. Springer, 2012.
- [46] Marc Lankhorst. *Enterprise architecture at work*, 2005.
- [47] O.I Lindland, G. Sindre, and A Solvberg. Understanding quality in conceptual modeling. *Software, IEEE*, 11(2):42–49, March 1994.
- [48] Pericles Loucopoulos, Peter McBrien, F Schumacker, Babis Theodoulidis, Vassilis Kopanas, and Benkt Wangler. Integrating database technology, rule-based systems and temporal reasoning for effective information systems: the tempora paradigm. *Information Systems Journal*, 1(2):129–152, 1991.
- [49] Raul Medina-Mora, Terry Winograd, Rodrigo Flores, and Fernando Flores. The action workflow approach to workflow management technology. In *Proceedings of the 1992 ACM conference on Computer-supported cooperative work*, pages 281–288. ACM, 1992.
- [50] George A Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J Miller. Introduction to wordnet: An on-line lexical database*. *International journal of lexicography*, 3(4):235–244, 1990.
- [51] D.L. Moody. The physics of notations: Toward a scientific basis for constructing visual notations in software engineering. *Software Engineering, IEEE Transactions on*, 35(6):756–779, Nov 2009.
- [52] C.W. Morris. *Foundations of the Theory of Signs*. International Encyclopedia of Unified science. University of Chicago Press, 1949.
- [53] Alexander Nossum and John Krogstie. Integrated quality of models and quality of maps. In *Enterprise, Business-Process and Information Systems Modeling*, pages 264–276. Springer, 2009.
- [54] J. L. H. Oei, L.J.G.T van Hemmen, E.D. Falkenberg, and S. Brinkkemper. The meta model hierarchy: A framework for information systems concepts and techniques, 1992.
- [55] Department of Defence (DoD). Dod architecture framework. version 1. volume i: Definitions and guidelines. *Office of the DoD Chief Information Officer, Department of Defense, Washington, D.C.*, 2003.
- [56] Hagelstein J. MacDonald L.G. Rolland C. Sol H.G. van Assche FJ.M. Verrijn-Stuart A.A. Olle, T.W. *Information Systems Methodologies: A Framework for Understanding*. Addison-Wesley, Reading, 1988.

- [57] Andreas L Opdahl and Brian Henderson-Sellers. Ontological evaluation of the uml using the bunge–wand–weber model. *Software and systems modeling*, 1(1):43–67, 2002.
- [58] Andreas L Opdahl and Guttorm Sindre. Facet modelling: an approach to flexible and integrated conceptual modelling. *Information Systems*, 22(5):291–323, 1997.
- [59] Jeffrey Parsons. An information model based on classification theory. *Management Science*, 42(10):1437–1453, 1996.
- [60] Ken Peffers, Tuure Tuunanen, Charles E Gengler, Matti Rossi, Wendy Hui, Ville Virtanen, and Johanna Bragge. The design science research process: a model for producing and presenting information systems research. In *Proceedings of the first international conference on design science research in information systems and technology (DESRIST 2006)*, pages 83–106, 2006.
- [61] Ken Peffers, Tuure Tuunanen, Marcus A Rothenberger, and Samir Chatterjee. A design science research methodology for information systems research. *Journal of management information systems*, 24(3):45–77, 2007.
- [62] Matti Rossi and Sjaak Brinkkemper. Complexity metrics for systems development methods and techniques. *Information Systems*, 21(2):209–227, 1996.
- [63] August-Wilhelm Scheer. *ARIS, Business Process Framework*.
- [64] Reinhard Schuette and Thomas Rotthowe. The guidelines of modeling—an approach to enhance the quality in information models. In *Conceptual Modeling—ER’98*, pages 240–254. Springer, 1998.
- [65] John R Searle. *Speech acts: An essay in the philosophy of language*, volume 626. Cambridge university press, 1969.
- [66] Yoav Shoham. Agent oriented programming: An overview of the framework and summary of recent research. In Michael Masuch and László Pólos, editors, *Knowledge Representation and Reasoning Under Uncertainty*, volume 808 of *Lecture Notes in Computer Science*, pages 123–129. Springer Berlin Heidelberg, 1994.
- [67] Keng Siau. Information modeling and method engineering: a psychological perspective. *Successful software reengineering*, page 193, 2002.
- [68] Keng Siau and Matti Rossi. Evaluation techniques for systems analysis and design modelling methods – a review and comparative analysis. *Information Systems Journal*, 21(3):249–268, 2011.
- [69] Keng Siau and Yuhong Tian. The complexity of unified modeling language: A goms analysis. *ICIS 2001 Proceedings*, page 53, 2001.
- [70] Bruce Silver. *Bpmn method and style, with bpmn implementer’s guide: A structured approach for business process modeling and implementation using bpmn 2*. 2011.
- [71] Richard C. Simpson. An xml representation for crew procedures. 2004.
- [72] J. F. Sowa and J. A. Zachman. Extending and formalizing the framework for information systems architecture. *IBM Syst. J.*, 31(3):590–616, June 1992.
- [73] R. K. Stamper. *Critical Issues in Information Systems Research*. John Wiley & Sons, Inc., New York, NY, USA, 1987.
- [74] Juha-Pekka Tolvanen, Matti Rossi, and Hui Liu. Method engineering: current research directions and implications for future research. In *Method Engineering*, pages 296–317. Springer, 1996.
- [75] Chris Tomlinson and Mark Scheevel. Object-oriented concepts, databases, and applications. chapter Concurrent Object-oriented Programming Languages, pages 79–124. ACM, New York, NY, USA, 1989.
- [76] Yi-Fu Tuan. Place: an experiential perspective. *Geographical Review*, pages 151–165, 1975.

- [77] P van Bommel, SJBA Hoppenbrouwers, HA Proper, and T van der Weide. Qomo: A modeling process quality framework based on sequal. In *Proceedings of EMMSAD*, volume 7. Citeseer, 2007.
- [78] R Hevner von Alan, Salvatore T March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *MIS quarterly*, 28(1):75–105, 2004.
- [79] Y. Wand and R. Weber. An ontological model of an information system. *Software Engineering, IEEE Transactions on*, 16(11):1282–1292, Nov 1990.
- [80] Yair Wand and Ron Weber. On the ontological expressiveness of information systems analysis and design grammars. *Information Systems Journal*, 3(4):217–237, 1993.
- [81] Yair Wand and Ron Weber. Research commentary: information systems and conceptual modeling—a research agenda. *Information Systems Research*, 13(4):363–376, 2002.
- [82] Yair Wand and Ron Weber. Reflection: Ontology in information systems. *Journal of Database Management*, 15, iii-vi, 2004.
- [83] Colin Ware. *Information visualization: perception for design*. Elsevier, 2013.
- [84] Timo Wegeler, Friederike Gutzeit, Aurèle Destailleur, and Bernhard Dock. Evaluating the benefits of using domain-specific modeling languages: an experience report. In *Proceedings of the 2013 ACM workshop on Domain-specific modeling*, pages 7–12. ACM, 2013.
- [85] George Wilkie. *Object-oriented software engineering: the professional developer's guide*. Addison Wesley Longman Publishing Co., Inc., 1993.
- [86] Klaus Wimmer. Conceptual modelling based on ontological principles. *Knowl. Acquis.*, 4(4):387–406, December 1992.
- [87] Terry Winograd and Fernando Flores. *Understanding computers and cognition: A new foundation for design*. Intellect Books, 1986.
- [88] AT Wood-Harper and Guy Fitzgerald. A taxonomy of current approaches to systems analysis. *The Computer Journal*, 25(1):12–16, 1982.
- [89] M. Yang. *COMIS - a conceptual model for information systems*. PhD thesis, IDT, NTH, Trondheim, Norway, 1993.
- [90] John W. Young, Jr. and Henry K. Kent. An abstract formulation of data processing problems. In *Preprints of Papers Presented at the 13th National Meeting of the Association for Computing Machinery*, ACM '58, pages 1–4, New York, NY, USA, 1958. ACM.
- [91] E.S.K. Yu and J. Mylopoulos. Using goals, rules, and methods to support reasoning in business process reengineering. In *System Sciences, 1994. Proceedings of the Twenty-Seventh Hawaii International Conference on*, volume 4, pages 234–243, Jan 1994.
- [92] John A Zachman. A framework for information systems architecture. *IBM systems journal*, 26(3):276–292, 1987.
- [93] M Zelm. Cimos: A primer on key concepts, purpose, and business value (technical report). *Stuttgart, Germany: CIMOSA Association*, 1995.