AUTOMATIC CONSTRUCTION OF CONCEPTUAL MODELS TO SUPPORT EARLY STAGES OF SOFTWARE DEVELOPMENT

- A SEMANTIC OBJECT MODEL APPROACH -

A THESIS SUBMITTED TO THE UNIVERSITY OF MANCHESTER FOR THE DEGREE OF DOCTOR OF PHILOSOPHY IN THE FACULTY OF ENGINEERING AND PHYSICAL SCIENCES

2015

By
Erol-Valeriu Chioașcă
School of Computer Science
Contents

Abstract 10
Declaration 11
Copyright 12
Acknowledgements 13
Publication List 16

1 Introduction 17
   1.1 Research Problem ........................................... 17
   1.2 Research Question .......................................... 19
   1.3 Research Aim and Objectives ............................... 19
   1.4 Research Contributions ..................................... 20
   1.5 Research Methodology ...................................... 21
   1.6 Overview of Thesis ......................................... 23

2 Background: Related Approaches 25
   2.1 Introduction ................................................. 25
   2.2 Analysis of Existing Approaches ............................ 26
      2.2.1 Linguistic-driven Approaches .......................... 26
      2.2.2 Pattern-based Semantic-driven Approaches .......... 29
      2.2.3 Ontology-based Semantic-driven Approaches ........ 33
   2.3 Summary ................................................... 36

3 Foundational Work 37
   3.1 Introduction ............................................... 37
   3.2 WordNet Verb Categories .................................. 38
**List of Tables**

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Semantic Object Models based on nine WordNet Verb Categories [Pri13, Fel90]</td>
<td>39</td>
</tr>
<tr>
<td>7.1</td>
<td>Quantitative results of the SOM validation experiment show the improvement in requirements completeness from Task 1 to Task 2</td>
<td>104</td>
</tr>
<tr>
<td>7.2</td>
<td>Qualitative results of the SOM validation experiment supports propositions P-1 and P-2</td>
<td>105</td>
</tr>
<tr>
<td>7.3</td>
<td>Approaches and examples evaluated</td>
<td>115</td>
</tr>
<tr>
<td>7.4</td>
<td>Correctness measures for related approaches and TRAM</td>
<td>116</td>
</tr>
<tr>
<td>7.5</td>
<td>Completeness measures for related approaches and TRAM (A - Action Completeness, T - Thing Completeness, R - Average Completeness for both Things and Actions)</td>
<td>117</td>
</tr>
<tr>
<td>7.6</td>
<td>Accuracy measures for related approaches and TRAM</td>
<td>119</td>
</tr>
</tbody>
</table>
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Research methodology adapted from [NJC90, MG95]</td>
<td>21</td>
</tr>
<tr>
<td>1.2</td>
<td>Roadmap - the arrows indicate the reading path between parts; absence of arrows indicate that there is no preferred sequence of reading.</td>
<td>24</td>
</tr>
<tr>
<td>2.1</td>
<td>The LOLITA core [Mic96]</td>
<td>27</td>
</tr>
<tr>
<td>2.2</td>
<td>Parse tree for sentence “John hit the ball”</td>
<td>27</td>
</tr>
<tr>
<td>2.3</td>
<td>Metamorphosis’ action transformation model [DPM05]</td>
<td>30</td>
</tr>
<tr>
<td>2.4</td>
<td>Architecture of the CIRCE environment [AG97]</td>
<td>32</td>
</tr>
<tr>
<td>2.5</td>
<td>The CM-Builder Approach [HG00]</td>
<td>34</td>
</tr>
<tr>
<td>3.1</td>
<td>An ER diagram for the example demonstrating the first four identity rules [Che83]</td>
<td>44</td>
</tr>
<tr>
<td>3.2</td>
<td>An ER diagram for the example demonstrating the final two identity rules [Che83]</td>
<td>45</td>
</tr>
<tr>
<td>3.3</td>
<td>Analysis and modelling process [RP92]</td>
<td>47</td>
</tr>
<tr>
<td>3.4</td>
<td>Mapping rules of OICSI to semantic network</td>
<td>50</td>
</tr>
<tr>
<td>3.5</td>
<td>Graphical representation of the semantic network [RP92]</td>
<td>51</td>
</tr>
<tr>
<td>3.6</td>
<td>Scenario meta-model [WZW+13]</td>
<td>55</td>
</tr>
<tr>
<td>3.7</td>
<td>OSM hierarchy showing subclasses of the Object Containment Model [Sut02]</td>
<td>57</td>
</tr>
<tr>
<td>3.8</td>
<td>Object Sensing OSM with associated generic requirements [Sut02]</td>
<td>59</td>
</tr>
<tr>
<td>3.9</td>
<td>The sentence “A man reads a book” represented through Sowa’s conceptual graph.</td>
<td>59</td>
</tr>
<tr>
<td>4.1</td>
<td>Example of SOM notation. Circles represent Things, Boxes represent Actions, full lines represent mandatory elements, dotted lines represent optional elements.</td>
<td>62</td>
</tr>
<tr>
<td>4.2</td>
<td>SOM concepts and their groupings</td>
<td>62</td>
</tr>
<tr>
<td>4.3</td>
<td>SOM Metamodel</td>
<td>64</td>
</tr>
</tbody>
</table>
4.4 Change SOM .......................................................... 65
4.5 Creation SOM ...................................................... 66
4.6 Motion SOM ......................................................... 66
4.7 Contact SOM ......................................................... 67
4.8 Possession SOM ..................................................... 68
4.9 Stative SOM .......................................................... 70
4.10 Cognition SOM ....................................................... 70
4.11 Perception SOM ..................................................... 71
4.12 Communication SOM .............................................. 72

5.1 SOM Translation Rule Sets ......................................... 75
5.2 Example of SON representation of behaviour and constraints . 81
5.3 SOMi of the example statement .................................. 81
5.4 SOMi of the example statement .................................. 82
5.5 SOMi of the example statement .................................. 82
5.6 SOMi of the example statement .................................. 83
5.7 The SOM approach seen as a series of process steps and their outputs 85
5.8 SON for the Library Callan example. Double lined circles represent captured behaviour patterns that link specific SOMi; hexagons represent identification keys mapped to captured constraints that are linked to specific actions ........................................ 88
5.9 A fragment of Library SON depicting XOR behaviour between two actions .................................................. 90

6.1 Overview of TRAM software components (NLR Parser, SOM Constructor, SOMi Composer, NLR Generator) mapped over the SOM approach ............................................. 92
6.2 TRAM Architecture, showing its main software components (NLR Parser, SOM Constructor, SOMi Composer, NLR Generator), Control Flow(solid arrows) and Data Flow (dotted arrows) .......................... 93
6.3 XML structure for the Parsed NLR .................................. 95
6.4 XMI snippet of a SOMi .............................................. 96
6.5 Fragment of SOM Ecore Metamodel .............................. 97
6.6 SON Ecore Metamodel ............................................. 99

7.1 Domain expert evaluation process of SON Correctness ........ 110
7.2 Modelling expert evaluation process of SON Completeness .... 112
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.3</td>
<td>Applying the same evaluation process to related approaches</td>
<td>114</td>
</tr>
<tr>
<td>B.1</td>
<td>Class diagram constructed by Callan [Cal94, p. 171]</td>
<td>147</td>
</tr>
<tr>
<td>B.2</td>
<td>Class model produced by CM-Builder [HG00]</td>
<td>148</td>
</tr>
<tr>
<td>B.3</td>
<td>Class diagram constructed by Gelhausen [Gel10]</td>
<td>149</td>
</tr>
<tr>
<td>B.4</td>
<td>Class diagram extracted by RACE [IA10]</td>
<td>150</td>
</tr>
<tr>
<td>B.5</td>
<td>Class diagram extracted by Shinde et al. system [SBM12]</td>
<td>150</td>
</tr>
<tr>
<td>B.6</td>
<td>Class diagram constructed by Li et al. approach [LDP04]</td>
<td>151</td>
</tr>
<tr>
<td>B.7</td>
<td>Class diagram constructed via CRC approach [HW97, p. 187]</td>
<td>151</td>
</tr>
<tr>
<td>B.8</td>
<td>Class diagram extracted by CIRCE [AG06]</td>
<td>152</td>
</tr>
</tbody>
</table>
Abstract

The earliest stage of software development almost always involves converting requirements descriptions written in natural language (NLRs) into initial conceptual models, represented by some formal notation. This stage is time-consuming and demanding, as initial models are often constructed manually, requiring human modellers to have appropriate modelling knowledge and skills. Furthermore, this stage is critical, as errors made in initial models are costly to correct if left undetected until the later stages.

Consequently, the need for automated tool support is desirable at this stage. There are many approaches that support the modelling process in the early stages of software development. The majority of approaches employ linguistic-driven analysis to extract essential information from input NLRs in order to create different types of conceptual models. However, the main difficulty to overcome is the ambiguous and incomplete nature of NLRs. Semantic-driven approaches have the potential to address the difficulties of NLRs, however, the current state of the art methods have not been designed to address the incomplete nature of NLRs.

This thesis presents a semantic-driven automatic model construction approach which addresses the limitations of current semantic-driven NLR transformation approaches. Central to this approach is a set of primitive conceptual patterns called Semantic Object Models (SOMs), which superimpose a layer of semantics and structure on top of NLRs. These patterns serve as intermediate models to bridge the gap between NLRs and their initial conceptual models. The proposed approach first translates a given NLR into a set of individual SOM instances (SOMi) and then composes them into a knowledge representation network called Semantic Object Network (SON).

The proposed approach is embodied in a software tool called TRAM. The validation results show that the proposed semantic-driven approach aids users in creating improved conceptual models. Moreover, practical evaluation of TRAM indicates that the proposed approach performs better than its peers and has the potential for use in real world software development.
Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.
Copyright

i. The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the “Copyright”) and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii. Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made only in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.

iii. The ownership of certain Copyright, patents, designs, trade marks and other intellectual property (the “Intellectual Property”) and any reproductions of copyright works in the thesis, for example graphs and tables (“Reproductions”), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv. Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see http://documents.manchester.ac.uk/DocuInfo.aspx?DocID=487), in any relevant Thesis restriction declarations deposited in the University Library, The University Library’s regulations (see http://www.manchester.ac.uk/library/aboutus/regulations) and in The University’s policy on presentation of Theses.
Acknowledgements

I feel extremely fortunate to have had the opportunity to work with Dr. Liping Zhao, Prof. John Keane, Dr. Goran Nenadić, and Dr. Richard Neville on the ideas found in this dissertation, and I would like to thank them for their support. Liping agreed to take me on as a graduate student, shared her experience and knowledge with me, and encouraged me to push the boundaries of the work I was carrying out. Her continuous support, trust, and guidance shaped me immensely as a researcher and writer. John and Goran have brought a fresh perspective on my work and offered me invaluable feedback and suggestions that always kept me on my toes. Richard offered me the advice and tools necessary to keep my work under control instead of the work taking control of me.

I am grateful to Dr. Alvaro Fernandes, Dr. Ian Pratt-Hartmann, Dr. Rizos Sakellarious, and Prof. Allan Ramsay for serving as examiners on my yearly progress reviews. Alvaro and Allan have been sources of motivation and advice without whom I would have never reached the final stages of Manchester’s gruelling graduate programme.

Special thanks to Tony Curran and Tim Duncan from IT support for doing an excellent job under heavy stress. They helped set up and maintain the infrastructure necessary for development and communication. Also thank you, Susannah ‘Susie’ Hymas, Chris Calland, Richard Ward, and Gavin Donald from Student Support Office for helping me find the way through the administrative labyrinth. You, unsung heroes, make the world go round!

An EPSRC EU Doctoral Training Award and a School of Computer Science Kilburn Award have supported my research over three years. I would like to express my gratitude to Manchester University and School of Computer Science for all the opportunities that they have given me over the years.

I would like to thank all my friends and colleagues in the Software Systems Group. I can think of no other individual that I can work so well with other than Keletso Joel Letsholo, who has been extremely patient with me. He was alongside me with advice
when I had doubts, always calm under pressure, never rushing - a sea of tranquility. I hope we get the chance to work together in the future. Dr. James Naish and Dr. Kawther Saaedi have been my mentors within the department. I want to thank James and Kawther for sharing their experiences with me. It was easier for me to step on the path that they already paved. Special thanks to Dr. Vicente Garcia-Diaz from University of Oviedo, Spain, who helped me enormously with the transformation of the approach to a model driven paradigm.

Over these years, my friends outside the department have enriched me emotionally, intellectually, and helped make my graduate years so exciting. I would like to thank the following people in particular: Goran Dokić, Dr. Marek Mikuš, Andrei Rădoi, Ioan ‘Freddy’ Ciocoiu, Theodoros Kyriakides, Deana Jovanović, Mehul Goel. Special thanks to Keira Slater for having the courage to read through my thesis, proofing and annotating it, and supporting me through the final stages of the process.

Finally, I would like to thank my always loving family for their support and advice: Sema, Vali, Tizi, Laur, Ababay, and Mamaie. My family made countless sacrifices for me, and have provided me with guidance and encouragement. This dissertation is dedicated to them.
Vă iubesc
Publication List


- Liping Zhao, Keletso Letsholo, Erol-Valeriu Chioasca, Sandra de F. Mendes Sampaio, Pedro Sampaio: Can business process modeling bridge the gap between business and information systems? *SAC 2012*: 1723-1724
Chapter 1

Introduction

1.1 Research Problem

In many system development projects, information requirements are initially documented in natural language, and then modellers convert these natural language descriptions into conceptual models such as Unified Modelling Language (UML) models, Entity-Relationship (ER) models, Business Process Models (BPM), or other similar representations. These conceptual models serve two fundamental roles in software development. First, they act as communication tools between system developers and users during the process of identifying information requirements. Second, they are the starting point from which software design models are developed.

In order to construct a conceptual model, a modeller needs to use domain-dependent and model-dependent knowledge iteratively [RP92]. Domain-dependent knowledge is used during the analysis of the application domain. Analysis begins with observations of the problem domain, in order to abstract and create problem statements that identify and classify phenomena. Such observations occur in different formats at different times for example, stakeholder interviews, workshops, or investigating existing documentation. Since most of this documentation is in English or other natural languages, this documentation is usually incomplete and ambiguous. Consequently, it is difficult to decipher the contents of these documents into comprehensive conceptual models. Moreover, these documents may contain both functional and non-functional requirements. However, the present work is focused on functional requirements i.e. requirements that state observable behaviour.

Model-dependent knowledge is used during the conceptual modelling process. During this process useful information is extracted from the previously created problem
CHAPTER 1. INTRODUCTION

statements and properties and constraints of the phenomena are described using a con-
ceptual model. The modeller relies on her expertise to extract useful information and
represent it in a comprehensive conceptual model. Consequently, this process is time
consuming and error prone. Moreover, errors undetected at this stage become pro-
gressively more expensive to fix at later stages [Dav93]. Therefore, there is a critical
need for devising a systematic approach for converting natural language descriptions
of functional requirements (NLRs) into comprehensive conceptual models.

The majority of the approaches developed to support the analysis process are linguistic-
driven. These approaches perform analysis by focusing on the surface structure of sen-
tences i.e. they link syntactic elements such as nouns and verbs to conceptual elements
such as entities and relationships. For example, Chen was the first to develop a set of
rules for directly translating English sentences into ER models [Che83]. A total of 11
translation rules were presented, divided into two categories: lexical identity rules and
linguistic pattern rules. The lexical identity rules, explicitly point out the correspon-
dence between grammatical constructs and ER model elements. Common nouns yield
entity types, transitive verbs indicate relationship types, and adverbs are attributes for
relationships. The linguistic pattern rules, identify linguistic patterns in English and
map them to ER model structures. The sentence “There are 200 employees in the de-
partment” is equivalent with “The department has 200 employees”, which has a clear
mapping to an ER model. Later, Abbott suggested that programs can be designed from
NL descriptions [Abb83]. He proposed assigning nouns to classes, verbs to methods,
and adjectives to attributes. The development of the object-oriented method led re-
searchers to adopt these concepts; classes and attributes are identified with nouns or
noun phrases, operations and relationships with verbs or verb phrases [Boo86]. How-
ever, an issue with linguistic-driven approaches is that the surface structure of NLRs
is ambiguous and incomplete i.e. some textual elements are not represented, while
others are indirectly referenced. Therefore, linguistic-driven approaches are unable to
produce comprehensive conceptual models due to their inability to identify missing
elements during analysis of the input NLRs.

Semantic-driven approaches have the potential to address the limitations of linguistic-
driven approaches. In order to perform analysis, semantic-driven approaches enhance
the linguistic methods by including domain specific knowledge encoded in the form
of patterns or domain specific ontologies. Using pre-defined sources of knowledge of
a specific domain offers the possibility of knowing whether an NLR is incomplete or
not.
After analysing the current state of the art, two categories of semantic-driven approaches become apparent. The first category, pattern-based semantic-driven approaches, assume that input NLRs are complete and proceed to extract information accordingly [AG97, AG06, DSM05, DPM05]. But unless the NLRs are created with completeness in mind, maybe using some form of restricted natural language, it is highly unlikely that unrestricted NLRs will be complete and unambiguous. The second category, ontology-based semantic-driven approaches, attempts to mitigate the NLR incompleteness by assuming that knowledge sources about a particular domain are pre-existent and complete before attempting analysis. Similarly to the previous assumption, it is improbable that such complete knowledge bases exist [HG00]. Therefore, it becomes apparent that neither types of semantic-driven approaches are able to produce comprehensive conceptual models due to the NLR incompleteness issue.

1.2 Research Question

This thesis proposes to put forward a semantic-driven approach that will address the limitations of current state-of-the-art transformation approaches in dealing with incomplete NLRs. The proposed semantic-driven approach should be applicable to any domain NLRs (similar to linguistic-driven approaches) and should be able to handle incomplete descriptions (similarly to semantic-driven approaches). Moreover, no knowledge of the analysed domains would exist beforehand. Consequently, the proposed semantic-driven approach is based on the following principal components:

- a method to unequivocally categorize and identify functional requirements described in natural language sentences based on verb categories
- sound and complete constructs used to transform textual requirements into comprehensive conceptual models of the world

The key research question of this project was thus whether or not such a semantic-driven approach can produce comprehensive conceptual models following the analysis process.

1.3 Research Aim and Objectives

The aim of this research is to develop an NLP-based modelling method and tool to support the conceptual modelling process. The main thesis of this research is that this
aim can be achieved by creating a software development tailored mechanism that can extract necessary elements from the input NLR using linguistic patterns based on both syntactic and semantic information. To achieve this, the research has the following objectives:

1. To investigate existing techniques and tools for analysis process and use this investigation as a background for current research.

2. To develop a new analysis method to bridge the gap between NLRs and their target conceptual models. This method will consist of a set of primitive semantic object models (SOMs), a semantic object network (SON), and a set of translation rules. This method is called the “SOM method”.

3. To develop an analysis tool based on the SOM method. This tool will contain the following components: NLR Parser, SOM Constructor, SOM Composer, NLR Generator. This tool is called “Textual Requirements to Analysis Models” (TRAM).

4. To evaluate whether the SOM approach improves the manual analysis process by identifying missing or overlooked elements in the input NLRs, therefore, producing more complete conceptual models.

5. To evaluate the comprehensiveness of TRAM conceptual models through examples and comparison to related work.

1.4 Research Contributions

This research project aims to put forth the following contributions:

- The development of a novel translation method that contains the following components:
  - Semantic Object Models (SOMs) - requirements-centred primitive models that connect a coherent set of modelling concepts with a specific WordNet verb category, with the purpose of extracting useful information from NLRs for conceptual modelling. SOMs operate at sentence level. The linguistic patterns capture English syntactic and semantic structures, whereas the verb categories are used to semantically select the appropriate linguistic patterns based on the sense of the verb in the NLR sentences;
1.5. Research Methodology

Figure 1.1: Research methodology adapted from [NJC90, MG95]

- Semantic Object Networks (SONs) - concept-centred knowledge representation graph obtained from the composition of SOM instances. SONs operate at document level. SONs describe the information captured from the input NLRs and support conceptual modelling. SON depicts NLR captured concepts and their interactions;

- A set of translation rules and analysis algorithm - the rules for extracting information from the NLR and creating SOM instances, as well as composing SOM instances into SONs; the analysis algorithm identifies missing requirements within the input NLR;

- The development of a prototype software tool - Textual Requirements to Analysis Model (TRAM), specifically the subsystem that transforms the input NLR into a SON;

- A systematic validation approach with human in the loop - this validation approach uses both modelling and domain knowledge to validate the output models; moreover, the validation approach has been developed into a system in order to reduce human bias.

1.5 Research Methodology

Figure 1.1 illustrates the research methodology employed for the current research project. The following paragraphs briefly describe the research process stages and the issues that need to be addressed in each stage.
Identifying the Research Problem

In this initial stage, research problems and related research questions are identified and their significance is justified. A clearly defined research problem provides a focus for the research throughout the development process. If the research proposes a new way of doing things, researchers have to develop a prototype to demonstrate the validity of the solution. Various other disciplines should be explored in addition to the existing research artefacts to find different ideas which could be incorporated in the new system. An ideal research problem is new, creative, and important [NJC90].

Developing the Software Prototype

This stage develops a software prototype in order to address the research problems identified. During prototyping, the following steps are followed:

- *Conceptual development* - during this phase a conceptual framework is developed in order to discuss the research questions. This framework also allows the investigation of functionalities and requirements of the system under development. Moreover, it facilitates the understanding of the building process.

- *Architecture development* - during this phase the system components are organised, the system functionality is defined, the static relationships and dynamic interactions between system components are established.

- *Analysis/Design* - during this phase the design of data structures, databases, or knowledge bases, as well as the specification of the program modules and functions is determined.

- *Implementation* - during this phase insights can be gained into the advantages and disadvantages of the used concepts, the frameworks, and chosen design. The experience is useful in re-designing the system.

Evaluating the Results

During this final stage, the system’s performance is tested to verify if it addresses the research problems. The results should be interpreted and evaluated based on the conceptual framework and design specification. Furthermore, evaluation results can be used to inform the further development of the system prototype or revision of the research problem. If any revisions are performed another evaluation is carried out. The
cycle is repeated until the research problems are satisfied. This research methodology yields several possible research contributions.

1.6 Overview of Thesis

This thesis has been organised into eight chapters as follows:

- **Chapter 2. Background: Related Approaches** presents a systematic literature review of existing approaches that transform NLRs into requirements models. From the analysis of the literature, research gaps are identified, prompting the need to address these gaps through a new approach.

- **Chapter 3. Foundational Work** describes seminal works that underpin the research efforts carried out during this project. These works are informed from multidisciplinary sources such as cognitive sciences, linguistics, and computer science.

- **Chapter 4. SOM Patterns** introduces and describes in detail the concepts and primitive models which are central to the approach described in this project. SOMs represent the main conceptual contribution of this research project.

- **Chapter 5. SOM Translation Rules and Application** presents the syntactic, semantic, composition, and NL generation rules necessary for the transformation of NLRs into requirements models through SOMs and their validation. Moreover, this chapter describes the process of the SOM approach through application of the translation rules on an example NLR, from the initial to the final step.

- **Chapter 6. Tool Support for the SOM Approach: The TRAM Architecture** gives an overview of the tool that support the SOM transformation approach through the perspective of its software architecture. Each component and its implementation is discussed. Each step of the process is described in the context of the software component carrying out the task.

- **Chapter 7. Validation and Practical Evaluation** evaluates the proposed transformation approach through experiments and direct comparisons against similar approaches. A comprehensive evaluation method was specifically developed for the evaluation process. All output requirements models are individually evaluated and compared. Moreover, a discussion of the results is included in this chapter, as well as an analysis of the comparison against related work.
Figure 1.2: Roadmap - the arrows indicate the reading path between parts; absence of arrows indicate that there is no preferred sequence of reading.

- *Chapter 8. Conclusions* ends the thesis by summarising the contributions and challenges of the research project. Moreover, limitations of the current work are discussed as well as directions for future work.

Figure 1.2 illustrates the structure of the thesis and recommended reading path.
Chapter 2

Background: Related Approaches

2.1 Introduction

Conceptual models are important because of the role they play in software development. They support communication between technical and non-technical stakeholders and they represent the starting point from which software design models are developed. A large volume of research has been carried out throughout the last 40 years to investigate automated translation of NLR to conceptual models. Through an analysis of these approaches, two categories have been defined based on (1) how these approaches deal with NLRs and (2) how these approaches perform NLR transformations.

14 existing approaches that deal with NLR transformations have been investigated. Following an analysis of these approaches it was established that linguistic-driven approaches make up the majority of the current state-of-the-art. More specifically, 11 out the 14 studied approaches were linguistic-driven. Their goal is to use the syntactic structure of the NLR sentences in order to extract textual elements and transform them into conceptual elements. Semantic-driven approaches, use pre-defined knowledge structures in order to extract textual elements from NLRs. These pre-defined structures can be modelled in the form of patterns or ontologies and could be potentially used to identify missing information from input NLRs. Therefore, two categories of semantic-driven approaches have been identified: pattern-based and ontology-based semantic driven approaches.

It is worth noting that the details of these approaches are not included in this section. However, the analysed approaches will be adequately discussed in the context of the defined categories i.e. how these approaches deal with NLRs (linguistic analysis) and how these approaches perform NLR transformations (using either syntactic
structures or semantic structures).

### 2.2 Analysis of Existing Approaches

#### 2.2.1 Linguistic-driven Approaches

Linguistic-driven approaches focus on extracting textual elements and assigning them conceptual semantic labels. This role assignment takes place by leveraging the relation between linguistic syntax and modelling semantics (e.g. nouns can be transformed into classes, verbs to methods, and adjectives to attributes).

The advantage of these approaches is their broad scope. They can be applied to any domain as long as the input NLRs are written in a language that can be parsed. The disadvantage is that these approaches can only extract information which is explicitly present in the NLR. However, in reality, NLRs rarely contain complete pieces of information. Therefore, linguistic-driven approaches cannot identify missing textual elements from NLRs.

Most of the NLR transformation approaches fall into this category [OLR01, IO06, YBL10, FKM+07, GKC07, SP10, Fri10, IA10, LDP04, EVR11]. NL-OOPS [Mic96] is a representative approach that showcases the general components of linguistic-driven approaches:

- *linguistic analysis* (lexical, syntactic, and semantic\(^1\)) represents the necessary tools to extract information from the NLRs surface structures;

- *transformation rules* use the tools provided by linguistic analysis to identify and select the information necessary to build conceptual models.

**Case Study: NL-OOPS**

Natural Language - Object Oriented Production System (NL-OOPS) is an NLP-based CASE tool that supports natural language requirements analysis by extracting the objects and their associations for use in creating object models [Mic96]. It is a representative example of a comprehensive linguistic-driven approach.

NL-OOPS uses a language processor called LOLITA to process single natural language requirements texts. LOLITA is described as being a large scale NLP system,\(^1\)

\(^1\)Semantic analysis, in the context of linguistic analysis, represents understanding word senses as well as relationships between words of the NLR.
2.2. ANALYSIS OF EXISTING APPROACHES

Figure 2.1: The LOLITA core [Mic96]

which analyses natural language text and extract information into a semantic graph called SemNet. SemNet can be used to build different models depending on the chosen object oriented methodology.

Requirements pre-processing places the textual requirements into a common data structure to enable further computational processing. LOLITA has a pipeline structure which allows it to analyse NL documents morphologically, syntactically, semantically, and pragmatically. The pipeline is implemented in a rule-based manner [MGC+95]. The steps involved in this process are:

- Lexical analysis - converts a sequence of characters into a sequence of tokens. A token, is a sequence of characters that has a certain meaning such as a word. Tokens are obtained by running the text through a tokeniser. The tokeniser splits the text into very simple tokens such as numbers, punctuation and words of different types. Next, the sentence splitter has the role of grouping tokens into sentences. The splitter uses lists of abbreviations to help distinguish sentence-marking full

Figure 2.2: Parse tree for sentence “John hit the ball”
stops from other kinds. Then, the POS tagger produces a part-of-speech tag such as “noun”, “verb”, “adjective” as an annotation on each word or symbol [Hep00]. Finally, morphological analysis returns the root and suffix of each word.

- **Syntactic analysis** - produces parse trees for each encountered sentence, showing the relationships between the constituent tokens. Usually, the parsers contain probabilistic models and follow the rules of a formal grammar. For example the sentence: “John hit the ball” has the following parse tree (Figure 2.2) where: $S$ represents the top level structure, $NP$ represents noun phrases, $VP$ represents verb phrases, $N$ represents nouns, $V$ represents verbs, and $DT$ represents determiners.

- **Semantic analysis** - determines if a node corresponding to a textual element already exists, if and how to build a new node, and how to connect the existing and new portions of the network [LG94]. This process takes place in three stages. First, all references such as pronouns or temporal adverbs are resolved (e.g. the pronoun “I” is replaces by a reference to the person that makes the utterance). Second, the grammatical parse tree is disambiguated (e.g. in the sentence “I like the bottle of wine on the table because of its fruity taste”, the pronoun “it” refers to the content of the bottle and not to the table). Third, new objects and their corresponding events are created.

- **Pragmatic analysis** - determines whether the new or modified portion of semantic network can be consistently combined with the existing network [LG94]. For example, in the sentence “I bought a car from the Japanese manufacturer Ford”, although syntactically and semantically correct, pragmatic analysis will identify that Ford is not a Japanese company using inference techniques. When the analysis produces a new piece of information, that knowledge is added to the SemNet as a disambiguated logical representation of the input. The implementation is rule-based, developed mostly in Haskell. SemNet is a semantic graph which contains a large number of nodes. 70% of SemNet's information comes from WordNet [MGC+95]. Thus, SemNet is heavily used in most stages of analysis performed by LOLITA.

The transformation occurs in two phases: (1) the context independent phase and (2) the context dependent phase.

During the context independent phase, NL-OOPS produces a list of candidate object classes, based on newly created SemNet nodes as well as previously existent ones.
In order to obtain the candidate object classes, there are certain rules that prescribe which nodes should be deleted or marked, based on the type of knowledge a certain node displays.

In the context dependent phase, object classes and associations are selected from the candidate list. The rules needed to select these elements are based on LOLITA’s events classification. The main steps of this phase are: events analysis (events are associated to candidate object classes), object identification (add attributes and methods to object classes), and association and multiplicity identification.

The output is an object model which contains classes, attributes, methods, associations, and multiplicities.

### 2.2.2 Pattern-based Semantic-driven Approaches

Pattern-based semantic-driven approaches focus on identifying pre-defined analysis patterns in the NLRs. In order to select a specific pattern, each element of the pattern needs to be identified in the input sentence. Once the pattern is identified, the textual elements are assigned conceptual semantic labels according to the pattern.

The advantage of these approaches is constituted by the fact that the patterns have their own structure and behaviour pre-defined. If the requirements in the NLRs are underspecified, the conceptual output models will contain comprehensive static and dynamic information from the pre-defined patterns. Unfortunately, the current pattern matching processes work only if all constituent pattern elements are identified in the text. Therefore, missing textual elements cannot be currently identified by these approaches.

Out of the 3 semantic-drive transformation approaches studied, Metamorphosis [DPM05] and CIRCE [AG97] are both pattern-based semantic-driven approaches. The following components are shared by both these approaches:

- *set of patterns* that encode domain specific knowledge, which represent the necessary tools to extract information from the NLRs surface structures;

- *transformation actions* use the pre-defined patterns to identify and select the information necessary to build conceptual models from NLRs.

The difference between Metamorphosis and CIRCE is that the former has more abstract patterns, therefore, is more general than the latter i.e. CIRCE needs to create patterns for every system specific term used in the NLR. Consequently, CIRCE is more
precise in its transformations, however, Metamorphosis has a more general scope of application.

Case Study: Metamorphosis

Metamorphosis is a pattern-based semantic-driven approach that facilitates the modelling of interactions and establishes persistent relationships between its input (Use Case Models) and output (Metamorphosis Conceptual Models). This approach is independent of system domains, technological aspects, and software development methods [DPM05, DSM05].

Metamorphosis (Figure 2.3) accepts as inputs Use Case Models written in natural language [DPM05, DSM05]. This input model describes a comprehensive and organised sequence of actions that are carried out by the actors during system interaction in order to achieve some goal. However, some simple guidelines are offered for writing the use cases [DSM05]:

- sentences must be active, affirmative, and declarative
- each sentence must have a single subject and a single main verb
- the main verb of the sentence must be transitive
- the sentence terminology must be normalised (e.g. control the use of synonyms)

Metamorphosis performs syntactic analysis and bases its linguistic approach on the action concept. Within the approach, actions can express (1) communication between actor and system, (2) internal system behaviour, or (3) control such as conditions, additions, repetitions, or groups of the aforementioned.

---

**Figure 2.3: Metamorphosis’ action transformation model [DPM05]**
The Metamorphosis Conceptual Model is represented by an Object Model and an Interaction Model. The Object model represents the structural aspects of a system, whereas the Interaction Model describes its dynamic behaviour. Thus, in the sense of the current review methodology, Metamorphosis is a direct transformation approach.

The transformation is based on Action Transformation Patterns (ATPs) that specify how actions are to be transformed into Object and respectively Interaction models. The patterns are domain and implementation independent.

In order to apply the ATPs, first roles of the textual elements need to be identified. Roles specify the properties that elements have within ATPs. Currently, these roles are assigned manually. The first step in applying ATPs is to recognize the pattern that must be applied. This is done by applying a formula which transforms the sentence into a formal representation, which is then compared against the formal representation of ATPs. When a match is found the respective pattern will be linked to that particular sentence. The second step is to obtain the structural and interaction fragments. Each ATP has a set of transformation rules, as well as a dynamic (Interaction Model) and structural model (Object Model) fragment. These rules are used to instantiate the two models with information from the text.

After all the sentences have been analysed, the resulting models need to be integrated. This process must guarantee the consistency and completeness of both Object and Interaction Models. In order to accomplish this, two rules have been presented:

- elements that are not yet present in the final conceptual model are added to the conceptual model with all their properties
- elements that are present in the final conceptual model must be checked to guarantee that the properties of these elements are kept

Case Study: CIRCE

CIRCE necessitates several elements put in place before the analysis process begins. First, the input requirements need to be completely and formally defined. Second, there need to be comprehensive patterns that capture the behaviour described in the NLRs. Third, matching mechanisms need to exist, that identify the required patterns.

CIRCE is an environment that supports modelling and analysis of requirements expressed in NL. The modelling and analysis activity is performed by a modular expert system called Cico. Cico recognizes NL sentences, extracts facts and transforms them
into abstract requirements, which are then passed to other tools that handle graphical representation, metrification, and analysis [AG97].

CIRCE takes as input requirements described in NL as well as a glossary that describes, classifies, and provides definitions of all the domain and system specific terms used in the requirements. The input requirements are written in NL, however, they have to be complete (i.e. each individual requirement must be fully understandable) and void of implicit references (e.g. lack of co-referencing pronouns). The glossary is built manually by the user and must contain a list of significant terms used in the requirements, which are classified using tags. Each term can be accompanied by a list of synonyms. The glossary contains structured information and must be written using a restricted set of syntactic conventions (e.g. simple, declarative, and complete sentences written in active voice).

Once the input is in place (both requirements description and glossary), CIRCE performs a series of textual transformations such as the substitution of glossary-defined synonyms and the simplification of composed words. Additionally, CIRCE carries out lexical and syntactic analysis by reducing typographical and formatting details to canonical forms and identifying parts of speech. Finally, the tokens are enriched with the semantic tags obtained from the provided input glossary and existing predefined glossaries and linked to any existing synonyms [AG06].

CIRCE performs the transformation of requirements to abstract requirements through a fuzzy matching process using a set of rules. These rules have three components: Model, Action, and Substitution. Thus, they are called the MAS rules. The MAS rules are defined at user level with the following semantics: $r = <m, a, s>$. The following
statement describes the application of such a rule: if a fragment of requirement matches
$m$, action $a$ is executed and the matching fragment of the requirements is replaced by $s$
[AG97].

CIRCE outputs various types of models with different purposes: model rendition
(Data Flow Diagram, ER, Object Oriented Paraphrase, Communication Diagram, De-
pendence Diagram), reporting views (Data Flow Diagram consistency check, Tag us-
age, ambiguity, and redundancy), metric views (Modified Feature Point, Understand-
ing Report, Residual abstractions). These models are output by view modules, which
take abstract requirements as input and output the necessary code representing the spe-
cific view. The transformation from abstract requirements to any of the conceptual
output models is performed using a rule-based approach.

2.2.3 Ontology-based Semantic-driven Approaches

Ontology-based semantic-driven approaches focus on building domain specific ontolo-
gies or discourse models using information contained in the input NLRs [HG00]. The
construction process starts from an initial simple, generic world model, to which NLR
extracted instances are added. This ontology model acts as an intermediate models that
is further transformed into a conceptual model. The advantage of these approaches is
that they can be used to identify missing concepts. The more information is present in
the initial world model, the better the approach is at identifying missing information. In
fact, a complete world model would contain the final conceptual model needed to de-
scribe the analysed system. However, it is highly unlikely that a complete world model
can in fact be created prior to analysis or otherwise [Har00]. Consequently, missing
textual elements cannot be systematically identified from input NLRs i.e. some ele-
ments may be identified, while others not, depending on the comprehensiveness of the
ontology used.

CM-Builder [HG00] is the only ontology-based semantic-driven approach. The
following items represent a high-level view of its components:

- a pre-defined ontology that encodes domain specific knowledge about the do-
  main to be analysed;

- a discourse interpreter that analyses individual sentences and adds new elements
to the pre-defined ontology;

- set of heuristics that build conceptual models using the information contained in
  the extended ontology.
Case Study: CM-Builder

CM-Builder is a NL-based CASE tool that supports the analysis stage of software development in an Object-oriented framework [HG00]. CM-Builder employs NLP techniques to analyse textual requirements written in English and build a discourse model of the processed text. The discourse model is used to automatically construct an initial UML Class Model representing the object classes mentioned in the text and their relationships.

CM-Builder takes as input a plain text file which contains software requirements written in English. There are no restrictions on the format. The document may contain a general problem statement describing the software or a list of functional requirements.

CM-Builder performs lexical, syntactic, and semantic analysis. From a lexical perspective it carries out tokenization, sentence splitting, POS tagging, and morphological analysis. From syntactic and semantic perspective it generates a parse tree using a feature-based phrase structure grammar and creates dependency relationships between the tokens. The analysis is carried out per individual sentences.

A specialised module, called the discourse interpreter, analyses the representations of each sentence previously produced and adds it to a predefined “world model” in order to extend it and produce a final discourse model of the input text. The transformation is written in the XI knowledge representation language, which allows the definition of ontologies. For each sentence, the transformation has the following steps:
2.2. ANALYSIS OF EXISTING APPROACHES

- Adding Instances and Attributes to the World Model, using rules such as the ones described by Chen [Che83] and Abbott [Abb83] adding simple noun phrase instances and their attributes under the object node in the world model, and verb instances and their attributes under the event node.

- Presupposition Expansion refers to the addition of elements to the world model that are not explicitly described in the text, based on some heuristics, e.g. “A book can be borrowed” might cause the addition of an object of type person to the discourse model as the agent of the borrow event. During the presupposition expansion stage, the pre-defined knowledge base is examined for any presupposition attributes that may encode useful information such as semantic role information about verbs and verb classes in the world model. However, there is no systematic guideline on identifying and creating such useful information.

- Coreference Resolution refers to the merging of two instances into a single one in the discourse model.

- Consequence Expansion checks if any syntactic rules may be applied as a consequence of merging instances in the previous stage. This may lead to the update of the discourse model by removing or adding instances or attributes. For example, the logical subject and object of a succession type of event (e.g. retire, resign) will represent the person and post objects.

For the final transformation from discourse model to analysis model, a set of heuristics based on properties of language, language patterns, and frequency counts is followed. For example:

- nouns are considered candidate classes, non-copular verbs are considered candidate relationships

- Some sentence patterns (e.g. “something is made up of something”, “something is part of something” and “something contains something”) denote aggregation relations.

- Determiners are used to identify the multiplicity of roles in associations

- Candidate classes that have low frequency and do not participate in any relationship are discarded from the list.

The output is an initial class model, which consists of classes, attributes, relationships, and multiplicities.
2.3 Summary

This chapter investigated and analysed approaches relevant to the translation of NLRs into conceptual models. These approaches have been discussed in the light of a defined category of approaches in order to highlight the intended research problem. The chapter that follows moves on to consider the seminal works that represent the foundation of the current research.
Chapter 3

Foundational Work

3.1 Introduction

This chapter follows on from the previous chapter, which examines different categories of NLR transformation approaches, and presents the investigation of fundamental techniques and theories that underpin the current work carried out during this research. The current review is guided by the hypothesis that emerged following the literature review. Therefore, this chapter, explores the current theories, techniques, and tools, that may be used to develop a new semantic-driven transformation approach that could overcome the shortcomings of current approaches.

The envisioned semantic-driven approach (SOM approach) bridges the gap between informal English and formal conceptual models. Therefore, two major mechanisms need to be developed. First, a mechanism related to language and extracting information from language is needed. To this extent, verbs were used as central elements of structuring sentences. The inspiration for abstracting verbs through their senses and therefore reducing the complexity of English comes from the Domain Theory [Sut02]. WordNet’s verb taxonomy is used with this goal in mind, as an abstraction mechanism [Fel90]. This taxonomy, allows for a comprehensive view over software engineering related English sentences through nine verb categories. Each category has its own specific dynamic properties and Fillmore’s Case Grammar [Fil67], is used to further define the verb categories, by adding further structural details.

Second, a modelling mechanism for the extracted linguistic information is needed. Thus, Bunge-Wand-Weber’s (BWW) Representational Theory [WW95] provides the necessary ontology for the complete representation of objects of interest using conceptual models. Additionally, scenario meta-models were used to inform the coupling
of the two major mechanisms (language and modelling), as they propose methods of bridging the gap between textual scenarios and target models.

Finally, Sowa’s conceptual graph [Sow84] was used as a representation mechanism for the proposed semantic-driven approach.

3.2 WordNet Verb Categories

Verbs are considered to be the distinctive trait of the human communication system and that they represent the essential component of language analysis [Leh72]. Arguably, they are the most important lexical and syntactic category of language [Fel90]. All English sentences contain at least one verb, but not necessarily a noun. Some linguists have proposed linguistic models in which verbs occupy the central organising elements of the sentence [Fil67]. Verbs provide relational and semantic framework for their sentences and their predicate-argument structures designate specific syntactic structures. Within these structures, the existing nouns will have certain roles attached to them that give specific meanings to the sentences. This syntactic and semantic information is often attributed to the verb and stored in the speaker’s memory [Fel90].

The proposed semantic-driven approach uses verbs to structure a sentence in a way that is useful for information extraction. This is done by attaching a specific model to a specific verb. However, due to the high number of verbs, a method of distinguishing between verbs is needed. In order to reduce the complexity posed by high numbers of verbs, the SOM approach employs the WordNet verb taxonomy.

WordNet’s verbs are organised in semantically coherent lexical groups. Verbs are divided into 14 semantic domains and an additional file which contains semantically diverse static verbs. All the verbs contained in the first group denote events or actions, whereas the latter verbs refer to states. WordNet’s scope is general, thus, for the purpose of translating NLRs into conceptual models within the scope of software development, the following nine verb categories have been imported from WordNet. These categories are described below.

The **Change** category includes all verbs that have the meaning of change, e.g. change, alter, vary, modify. This is one of the largest verb categories in WordNet [Fel90].

The **Creation** category consists of verbs that refer to manners of creation such as *create by mental act* (invent, conceive, etc.); *create by artistic means* (engrave, illuminate, print); *create from raw material* (weave, sew, bake). Many of these verbs
3.2. WORDNET VERB CATEGORIES

Table 3.1: Semantic Object Models based on nine WordNet Verb Categories [Pri13, Fel90]

<table>
<thead>
<tr>
<th>Verb Category</th>
<th>Example of Verbs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change</td>
<td>renew, edit, extend, change, etc.</td>
</tr>
<tr>
<td>Creation</td>
<td>assemble, build, etc.</td>
</tr>
<tr>
<td>Motion</td>
<td>move, enter, etc.</td>
</tr>
<tr>
<td>Contact</td>
<td>shut off, put, turn off, etc.</td>
</tr>
<tr>
<td>Possession</td>
<td>get, has, borrow, buy, sell, own, etc.</td>
</tr>
<tr>
<td>Stative</td>
<td>verbs of having, spatial relations, etc.</td>
</tr>
<tr>
<td>Cognition</td>
<td>read, know, think, judge, analyse, etc.</td>
</tr>
<tr>
<td>Perception</td>
<td>observe, display, see, etc.</td>
</tr>
<tr>
<td>Communication</td>
<td>query, ask, issue, tell, order, etc.</td>
</tr>
</tbody>
</table>

can have a direct object (that is a key object) that refers to the creation.

The Motion category consists of verbs organized into two groups: (1) make a movement and (2) travel.

Contact is a category whose central verb concepts are fasten, attach, cover, cut, and touch. This is the largest verb category in WordNet [Fel90].

The Possession category consists of verbs mostly derived from three groups of basic concepts: (have, hold, own), (give, transfer), and (take, receive). This verb category denotes the change of possession and its prior or resultant state.

The Stative category mostly consists of verbs of being and having. This category is similar to the adjectives category, and therefore it is usually used to capture properties of Thing.

The Cognition category denotes various cognitive actions and states, such as reasoning, judging, learning, memorizing, understanding, and concluding.

The Perception category contains verbs referring to sensory perception, e.g. “the foreman observes the water level”.

The Communication category consists of verbs of verbal and non-verbal communication (e.g. gesturing). Verbal verbs include speaking and writing. The Communication verbs overlap to a large extent with the Cognition verbs, as one cognition verb can refer both to the mental activity and to the verbal communication.

These verb categories are summarised in Table 3.1.
CHAPTER 3. FOUNDATIONAL WORK

3.3 Case Grammar

Surface structure and deep structure are two types of representation of a sentence expressed in a natural language [Cho02]. Surface structure represents the syntactical form of the sentence, as described by grammatical rules. Deep structure describes the meaning that a sentence conveys. For example, “John printed a report” and “The report was printed by John” have different surface structures but the same deep structure. Consequently, there is one deep representation that captures multiple but identical in meaning surface structures of a sentence.

Case grammar is a conceptual framework developed by Fillmore that describes the deep structure of a sentence using the concept of case [Fil67]. The author posits that each simple sentence consists of a verb and one or more noun phrases, each associated with the verb in a particular case relationship. Furthermore, each case relationship appears only once, except for the situations in which same relationship type cases are composed using conjunctions.

The case grammar lends to the SOM approach the idea that every word in a sentence has a specific case (or role). However, the SOM approach takes Fillmore's idea a step further by positing that identical role models can be used to capture different sentences. For example, the same model of transfer of possession can be used to model sentences of “buying” as well as those of “selling”.

The case notion contains a set of inherent concepts that are naturally used by humans to describe situations (e.g. who performed a certain action, what changed, who was affected by an action etc.). The following cases have been proposed by the Fillmore:

- **Agentive** (A) - the case of the typically animate perceived instigator of the action identified by the verb.
- **Instrumental** (I) - the case of the inanimate force or object causally involved in the action or state identified by the verb.
- **Dative** (D) - the case of the animate being affected by the state or action identified by the verb.
- **Factitive** (F) - the case of the object or being resulting from the action or state identified by the verb, or understood as part of the meaning of the verb.
- **Locative** (L) - the case which identifies the location or spatial orientation of the state or action identified by the verb.
3.3. CASE GRAMMAR

- **Objective** (O) - the semantically most neutral case, the case of anything representable by a noun whose role in the action or state identified by the verb is identified by the semantic interpretation of the verb itself; conceivably the concept should be limited to things which are affected by the action or state identified by the verb. The term is not to be confused with the notion of direct object, nor with the name of the surface case synonymous with accusative.

Cases can be used to classify verbs (e.g. by defining sentence types through case associations), as well as to explain various co-occurrence restrictions. For example, in 1 the subject “John” is in an Agent relationship with the verb; in 2 the subject “hammer” is an Instrument; and both Agent and Instrument appear in sentence 3.

- John broke the window - the subject is also an Agent (“John”)
- A hammer broke the window - the subject is also an Instrument (“hammer”)
- John broke the window with the hammer - both Agent (“John”) and Instrument (“hammer”) appear in this sentence

The fact that “John” and “hammer” are different cases makes impossible the composition of the two in a sentence such as (1). Moreover, the restriction that only one type of case relationship can be present in a sentence is apparent in (2).

(1) John and a hammer broke the window.
(2) A hammer broke the glass with a chisel.

Within the case grammar framework, a sentence is described as such

\[
S \rightarrow M + P
\]

where \(P\) represents a tense-less set of relationships involving verbs and nouns and can be expressed as

\[
P \rightarrow V + C_1 + \ldots + C_n.\]

On the other hand, \(M\) contains elements such as negation, tense, mood, and aspect. The cases cannot be easily mapped to the surface structure relations. For example, in 3 and 4 “John” is A; in 5, 6, and 7 the “key” is I; in 8, 9, and 10 “John” is D; finally, in 11 and 12 “Chicago” is L.
(3) John opened the door.

(4) The door was opened by John.

(5) The key opened the door.

(6) John opened the door with the key.

(7) John used the key to open the door.

(8) John believed that he would win.

(9) We persuaded John that he would win.

(10) It was apparent to John that he would win.

(11) Chicago is windy.

(12) It is windy in Chicago.

The concept of case frame describes the case environment that the sentence provides i.e. the array of case relationships contained in the sentence. According to the case grammar conceptual framework, the case frame selects the verb. For example, verbs like “run” may be inserted in the frame [__A], verbs like “sad” into the frame [__D], verbs like “remove” and “open” into [__O+A], verbs like “murder” and “terrorize” into [__D+A], verbs like “give” into [__O+D+A].

Fillmore suggests “frame features” as elements that indicate the set of case frames into which a certain verb may be inserted. This in essence represents a verb classification mechanism. However, the classification is non trivial because of the high number of possible case relationship combinations, as well as the fact that one verb may occur in multiple distinct case arrays. For example, the verb “open” can appear in the following frames:

- The door opened - [__O]
- John opened the door - [__O+A]
- The wind opened the door - [__O+I]
- John opened the door with a chisel - [__O+I+A]
Thus, a comprehensive frame representation for the verb “open” would be \([\_\_O(I)(A)]\), where the round brackets indicate optional elements.

The issue with this method is that all senses of a verb need to be considered and appropriate frame representations need to be developed in a bottom-up approach. The solution currently proposed is to create model representations in a top-down manner, by considering verb senses first (which are extracted from WordNet) and then create model representations for each sense (following the case grammar method), therefore ensuring comprehensive coverage of English sentence structures.

### 3.4 ER Modelling

The ER modelling method represents a fundamental approach to conceptual modelling. The linguistic rules that have been initially described by Chen [Che76] have also been integrated in the SOM approach. In the following, an overview of ER modelling is presented.

The ER diagram has been proposed for database description by Chen [Che76]. It contains four types of concepts:

- **Rectangles** represent **entity types**
- **Diamonds** represent **relationship types**
- **Circles** represent **value types**
- **Lines** that connect rectangles (entity types) and diamonds (relationship types) to value types represent **attributes**.

In 1983, Peter Chen proposes 11 rules for translation of English sentences into Entity-Relationship (ER) diagrams [Che83]. These rules can be further divided into two groups: identity rules and pattern rules. Identity rules explicitly point out the correspondence between English grammatical constructs (i.e. nouns, verbs, adjectives, and adverbs) and ER model elements [Che83]:

1. A **common noun** (such as “person”, “chair”) corresponds to an **entity type** in an ER model.
2. A **transitive verb** corresponds to a **relationship type** in an ER model.
3. An **adjective** corresponds to an **attribute of an entity** in an ER model.
4. An **adverb** corresponds to an **attribute of a relationship** in an ER model.

5. The **objects** of algebraic or numeric operations can be considered as **attributes**.

6. A **gerund** corresponds to a **relationship-converted entity type** in ER models.

The following example was taken from Chen’s paper [Che83] to demonstrate the previous rules:

**Example for Rules 1 - 4**

*English Statement:* A 40-year-old person works on a project with project number 2175 for 20% of his time.

*Analysis:* “Person” and “project” are nouns and can be considered as entity types. Since “40-year-old” is an “adjective” modifying the noun “person”, “number of year old” (or “age”) can be considered as an attribute of person entities. Similarly, since “with Project number 2175” is an adjective phrase modifying the noun “project”, “project number” can be viewed as an attribute of “project” entities. “Works on” is a transitive verb phrase and therefore corresponds to a relationship type. Since “for 20% of his time” is an adverb phrase used to modify the verb phrase “works on”, “percentage of time” can be considered as an attribute of “works on” relationships.

*ER diagram:* The corresponding ER diagram is shown in Figure 3.1.
3.4. **ER MODELLING**

**Example for Rules 5 and 6**

*English Statement:* A minimum number of 7284 units of product are shipped to customers, and the shipping is performed by clerks.

*Analysis:* Since “minimum” is an algebraic operation, it may be inferred that “number” is an attribute of “product” entities. Since a gerund is a noun converted from a verb, and taking into account the rules 1 and 2, it may be deduced that gerunds correspond to entity types converted from relationship types.

*ER diagram:* The corresponding ER diagram is shown in Figure 3.2.

![Figure 3.2: An ER diagram for the example demonstrating the final two identity rules](image)

Pattern rules, identify patterns in English language syntax and map them to ER model structures:

1. If the sentence has the form “There are . . . X in Y”, then it can be converted into the equivalent form “Y has . . . X”. For example, “There are 200 employees in the department” is equivalent with “The department has 200 employees”.

2. If the sentence has the form “The X of Y is Z” and if Z is a proper noun, X may be treated as a relationship between Y and Z. In this case, both Y and Z represent entities. For example, “The father of James Smith is Robert Smith” can be represented in an ER model where James Smith and Robert Smith refer to entities of type “person”, and “father” is a relationship between the two entities.

3. If the sentence has the form “The X of Y is Z” and if Z is not a proper noun, X may be treated as an attribute of Y. In this case, Y represents an entity (or a
group of entities), and \( Z \) represents a value. For example, “The color of the desk is blue”, may be represented in an ER models where “desk” is an entity that has a “color” attribute.

4. A **clause** is a high-level entity type abstracted from a group of interconnected low-level entity and relationship types in ER models. For example, in the sentence “The managers decide which machine is assigned to which employee”, “which machine is assigned to which employee” is a clause that serves as direct object for the verb “decide”. Within this clause, “machine” and “employee” are entity types with an “assigned to” relationship type between them.

5. A **sentence** corresponds to one or more entity types connected by a relationship type, in which each entity type can be decomposed into low-level entity types interconnected by relationship types. For example, sentences may be decomposed into clauses, which in turn may be decomposed into sub-clauses, each containing entity and relationship type elements.

### 3.5 OICSI

Rolland and Proix present one of the first general linguistic approaches to support the generation of conceptual specification from input NLRs [RP92]. Moreover, they are the first to propose the use and adaptation of Fillmore’s cases to software engineering. Consequently, this work is considered fundamental in proposing an initial practical idea of how to integrate linguistic approaches in software development. In the following, an overview of this idea is presented.

Requirements engineering consists of tasks such as knowledge acquisition and validation. Figure 3.3 shows the two main processes during knowledge acquisition: analysis and modelling of relevant parts of the problem domain [RoI98]. First, the analyst observes phenomena in the problem domain and creates problem statements capturing the state of the domain during analysis. Second, the analyst represents and describes the observed phenomena by mapping the problem statements onto concepts of a specific conceptual modelling language.

Validation is the process during which the conceptual model created by the analyst is checked against the actual phenomena occurring in the problem domain. This process involves both the analyst and the domain expert.
Rolland and Proix propose a linguistic approach that aims to support the requirements engineering process in a way that supports the abstraction process. The authors propose to achieve this approach by formalising the cognitive behaviour of analysts during the acquisition phase of the requirements engineering process. Furthermore, the authors propose a validation mechanism that allows the specification to be diagnosed from the domain expert’s point of view.

Rolland and Proix use Chen’s rules as a starting point to define sentence patterns. Moreover, they describe the need to make explicit the sentence patterns and their different types. The authors propose OICSI [RP92], which is a system that attempts to identify different sentence patterns and classify their component elements based on Fillmore’s case system [Fil67].

OICSI implements a specialisation of Fillmore’s case system, which consists of two main modifications. First, cases can be applied to clauses, as opposed to individual words. In this case the analysis of a complex sentence is performed in a top-down fashion. Subordinate clauses are identified with a certain case based on the verb of the main clause. The following step is to apply the case approach within each of the subordinate clauses.

Second, case classification has been revised in order to reflect the structure of problem statements created during the analysis process. The authors propose a binary classification of problem statements: fact descriptions (e.g. The status of each copy of a book is recorded in real time) and rules (e.g. Subscription fees are paid every
CHAPTER 3. FOUNDATIONAL WORK

year). OICSI uses the following cases: OWNER, OWNED, ACTOR, TARGET, CONSTRAINED, CONSTRAINT, LOCALIZATION, ACTION, OBJECT.

For example, in the sentence “A subscriber borrows books”:

- **subscriber** - is associated to the ACTOR and OWNER cases
- **books** - is associated to the OWNED case
- **A subscriber borrows books** - is associated to the ACTION case

The linguistic patterns used by OICSI are based on the specialised cases and on a verb class hierarchy. The verb class hierarchy contains four classes of verbs: OWNERSHIP (e.g. to include, to have, to compose), ACTION (e.g. to make, to update, to record, to erase), STATE (e.g. to be, to appear), EMERGENCE (e.g. to arrive, to occur, to happen).

There are two types of patterns: elementary patterns and sentence patterns. Elementary patterns map syntactic units of a clause to a specific case and they are in turn of three types: structural, behavioural, and constraint patterns. These patterns specify tuples consisting of a syntactic unit (nominal groups (Ng), clause (Cl), subordinate clause (Sub), and main clause (Mn)) and the case associated with it (e.g. [Ng_subject](OWNER), [Ng_complement](OWNED)). There are two structural patterns:

- **SP1**: [Ng_subject](OWNER) [verbal form](ownership_subject) [Ng_complement](OWNED). For example: “any subscriber has a name and address”.
- **SP2**: [Ng_subject](OWNED) [verbal form](ownership_complement) [Ng_complement](OWNER). For example: “loan-requests are made by subscribers”.

There are four behavioural patterns:

- **BP1**: [Ng_subject](ACTOR) [verbal form](action) [Ng_complement] (TARGET). For example: “subscribers borrow books”.
- **BP2**: [Conjunction](LOCALIZATION) [Ng_subject](ACTOR) [verbal form](action) [Ng_complement](OBJECT). For example: “when a subscriber returns a book copy”.
- **BP3**: [preposition](LOCALIZATION) [Ng](OBJECT). For example: “as soon as the receipt of a subscriber’s subscription fees”
3.5. OICSI

- **BP4**: [Ng](TARGET) [verbal form](action). For example: “the loan is agreed upon”.

There is one constraint pattern:

- **CP1**: [Ng_subject](CONSTRAINED) [verbal form](state) [Ng_complement] (CONSTRAINT). For example: “the number of loans is equal or less than three”.

Sentence patterns map clauses of a sentence to specific cases. There are two sentence patterns:

- **SPT1**: [Main clause]. This pattern corresponds to sentences that have one main clause, which matches one of the three types of elementary patterns described above (structural, behavioural, constraint).

- **SPT2**: [Subordinate clause unifying a BP pattern](LOCALIZATION) [Subordinate clause unifying a BP2 pattern](CONSTRAINT) [main clause unifying a BP pattern with a verb expressing an action](ACTION + CONSTRAINED). For example: “When there is a loan request, the loan is agreed only if the subscriber’s status is “active” and if a copy of the requested book is available”.

The conceptual model generation starts from the assumption that the cases are already linked with the syntactic elements. The linguistic approach is independent of the conceptual models chosen for representation. The rules that map the cases to the target conceptual model are language dependent. In the current study, the authors map OICSI onto a semantic network which contains four types of nodes (entity node, event node, action node, constraint node) and five types of arcs:

- rl: relationship between two object nodes

- md: action modifies object

- tr: event triggers an action

- act: object changes state

- ct: node (object, action, event) has a constraint
Requirements Processing

OICSI performs lexical and syntactic analysis in order to process textual requirements. During the lexical analysis, each verb is classified according to one of the four classes (i.e. ownership, action, state, and emergence) using a dictionary. The syntactic analysis performs two roles. First, it validates that the analysed textual requirements are written in the expected language. Second, it produces parse trees.

Additionally, a linguistic step is performed, which matches the produced parse trees onto the defined sentence patterns and associates each syntactic unit with a case. The matching and association processes occur in the same step and are supported by linguistic rules. These rules are composed of two parts: a premise and a conclusion. The premise represents the condition which allows the recognition of the sentence or clause pattern. The conclusion associates cases to the sentence or clause elements. In order to achieve the pattern recognition the previously identified verb class as well as the grammatical structure of the sentence or clause are used.

The authors, provide two examples of linguistic rules for implementing the SP1 pattern.

**RL1:**

\[
\begin{align*}
\text{IF } \text{meaning(clause(verbal form))} &= \text{ownership_subject} \\
\text{AND } \text{gram_structure(Ng_subject)} &= \langle \text{article, noun}_1 \rangle \\
\text{AND } \text{gram_structure(Ng_complement)} &= \langle \text{article, noun}_2 \rangle \\
\text{THEN } \text{case(noun}_1) &= \text{OWNER} \\
\text{case(noun}_2) &= \text{OWNED}
\end{align*}
\]
3.5. OICSI

Figure 3.5: Graphical representation of the semantic network [RP92]

**RL2:**

IF meaning(clause(verbal form)) = ownership_subject
AND gram_structure(Ng_subject) = ⟨article, noun_1, predicate_1⟩
AND gram_structure(Ng_complement) = ⟨article, noun_2⟩
AND gram_structure(predicate_1) = ⟨preposition, article, noun_3⟩
THEN case(noun_1) = OWNER(verb)* and OWNED(predicate)
case(noun_2) = OWNED.

* the notation OWNER(verb) and OWNED(predicate) mean that the role OWNER is played in regards to the verb and that the role OWNED is played in regards to the predicate. By default, the case meaning is in regards to the verb.

**Transformation from Requirements to Semantic Network**

The processed requirements are transformed into the semantic network according to the mapping rules described in Figure 3.4. An example of the semantic network representation is depicted in Figure 3.5.
3.6 Modelling Concepts

So far, this chapter has described the foundational works and how they are relevant to the proposed SOM approach, specifically its linguistic mechanism. In the following, the foundational works that are pertinent to the modelling mechanism are presented.

The BWW ontology [WW90] was used as a guide in the construction of the SOM metamodel. This process guarantees that the SOM metamodel is able to generate models that are good representations of the real world. Moreover, scenario-based software development approaches were used to inform the coupling of linguistic and modelling mechanism, as they propose methods of bridging the gap between textual scenarios and target models [Car00, SMMM98, WZW+13].

Lastly, conceptual graphs are used to visually represent and support the extracted information [Sow84].

3.6.1 Information Systems Ontology

Ontological analysis lies at the core of knowledge representation and also defines the structure of knowledge in a given domain [CJB99]. Ontology is a branch of metaphysics and attempts to organise and describe reality in a structured approach. Jackson states that the problem is situated in the real world and we have to focus on observable physical phenomena in order to understand it. This will help us verify that we actually satisfy the requirements [Jac01]. Ontology contains concepts that describe the physical phenomena of the real world [WW93]. Ontologies, especially in RE, provide a neutral computational medium within which information from different sources (i.e. different modelling grammars) can be represented, kept and maintained [UG96]. Moreover, ontologies provide the means to precisely define terms used in various contexts and support the formation of logical connections between elements contained in the model [UG96].

The number of conceptual modelling techniques, with no theoretical foundation, used to define IS requirements have constantly been rising [RIRG05]. Concerned with this issue, Wand and Weber have developed a set of three models for evaluating conceptual modelling languages based on Bunge’s ontology: representational model, state-tracking model, and good-decomposition model [WW08].

In this project we focus on the representational model of the BWW framework
This model (or ontology) is concerned with establishing the strengths, weaknesses, completeness and efficiency of modelling grammars from an ontological viewpoint, as explained in [GR00]. BWW ontology is used as a reference for checking the completeness of conceptual languages in terms of their ability to generate models that are good representations of the real world. There are two evaluation criteria: completeness (if a language contains constructs that correspond to any BWW concept) and clarity (if each construct of a language has a one-to-one correspondence with one of BWW’s constructs).

In order to capture any real world phenomena pertaining to IS, the BWW ontology is comprehensive. Attempts have been made to create a metamodel of the BWW constructs, however, the results are highly complex and large for the scope of the current research [RGI04]. Fortunately, Wand and Weber suggest that the BWW ontology is not monolithic, but rather hierarchical [WW95]. There are five fundamental ontological constructs (Thing, Property, State, Transformation, and Stable State) from which all other constructs are derived from [WW90]. The SOM metamodel uses all five BWW fundamental constructs as a starting point for the scope of the current research, leaving room for further expansion if needed. Therefore, the SOM metamodel displays ontological completeness and clarity [WW08]. These fundamental concepts are perfectly adapted to the scope of the current research (i.e. analysis of natural language descriptions of functional requirements) [PA98, SK98]. Consequently, the SOM metamodel is sound and complete in capturing FRs described in NLRs.

There are several reasons for using BWW. First, it is aimed at information systems (IS) and has a formal specification. Second, some of the results obtained using BWW were empirically tested and supported the view that the model represents a good basis for analysing representational capabilities of conceptual modelling languages [Rec10, RIRG06]. Third, it is well spread in the process modelling domain and it is an upper ontology which allows for a wide scope and applicability. Lastly, it has achieved a certain level of maturity, adoption and dissemination in research during the last 20 years.

BWW is not without opposition [Wus06]. However, Wand and Weber promptly addressed these views [WW06] and currently there are no known open debates regarding BWW’s suitability as an established representational analysis method.

---

1. Thing, Property, and State have kept the same name in the SOM metamodel. Transformation has changed to “Action”, whereas Stable State is equivalent to “Goal”.
3.6.2 Scenario Meta-model

A further source of inspiration for the SOM metamodel is the scenario metamodel. The scenario metamodel offers insight into how to combine different ontological concepts that are relevant for some modelling task. More specifically, the scenario metamodel presented below [WZW+13] represented a starting point on how to couple BWW’s principal ontological concepts among themselves. The following presents a brief overview of scenario usage in software development and the scenario meta-model that acted as inspiration for the SOM approach.

Scenario usage is widespread within software engineering, in all phases of the process [WPJH98, dPLHDK00, Sut03]. Moreover, there are arguments that scenarios act as starting points for all modelling and design activities within software development [Sut03]. Within the RE phase scenarios have been used as a communication tool to bridge the gap between technical and non technical stakeholders during elicitation [HH82, HPW98], modelling [Kai00], and specification [SMMM98, Sut98].

There is a wide variety of scenario types [RAC+98], however, within the scope of the current research, scenarios are defined as textual descriptions of user and system behaviour [Car95, Her03]. Their narrative form imposes temporal structure to the textual specification document as well as make them an ideal vehicle for knowledge capture and communication.

However, scenarios are not without shortcomings such as being incomplete and implicit [Sut03], noisy [DLVL06], biased [Dia02], unstructured [Sow84]. Additionally, scenarios can become large, complex, and difficult to process and use [SMMM98]. Considering both the advantages and shortcomings of scenario usage in software development, it becomes apparent that scenarios can be complemented in order to leverage their assets and minimise their weaknesses. For example, both Sutcliffe [Sut03] and Damas et al. [DLVL06] propose using models to develop, structure, and represent requirements.

Based on the work of Zhao [Zha11], Wang et al. propose a refined version of Pattern Language for Transforming scenarios (PLANT) [WZW+13]. The purpose of PLANT is to bridge the gap between textual scenarios and their target models. The method through which this is accomplished, relies on four patterns: Establishing the Story Line, Elaborating Things that Change, Identifying Agents and Their Interactions, and Unravelling the Goal and its Sub-goals. Each one of these four patterns offers guidelines on how to transform certain parts of a scenario into a specific target model. Finally, the initial input scenario is represented by four interrelated output models.
3.6. MODELLING CONCEPTS

PLANT’s pattern concepts, pattern language, and rules are based on Alexander’s [Ale79, AIS77] work on defining architectural patterns. On the other hand, PLANT’s theoretical foundations are based on cognitive structures [Joh87, Lak90], and requirements engineering derived meta-models [RSA98, SMMM98, Mai98, ZMT05, Kai00]. The scenario meta-model obtained from analysing cognitive and requirements engineering structures is depicted in Figure 3.6.

According to PLANT, a scenario is structured by the SOURCE-PATH-GOAL schema [WZW+ 13], where:

- **SOURCE** contains the initial state that defines a precondition so that a scenario may be triggered
- **PATH** contains a sequence of actions
- **GOAL** contains the final state that defines the outcome that is achieved after a scenario is triggered

A scenario may be described as being normal or abnormal [RSA98, SMMM98]. During the normal scenario the desired outcome is obtained, whereas during the abnormal scenario an outcome that should be avoided is obtained. Therefore, there are two types of goals defined: the attainment goal (described in the normal scenario) and the avoidance goal (described in the abnormal scenario) [VL01]. Additionally, there are one or more agents involved in the scenario, carrying out some actions, either individually or interactively, in order to achieve the desired goal state. Objects change

Figure 3.6: Scenario meta-model [WZW+13]
their states under the effects of the actions performed.

### 3.6.3 Domain Theory

The main idea that the Domain Theory inspired in the SOM approach is the creation of a hierarchy of models that describe behaviour (i.e. the SOMs). This hierarchy is created bottom-up through abstraction, therefore allowing for reduction of complexity, but preservation of enough functional details so that the models are still practical. The following presents an overview of the Domain Theory.

The Domain Theory proposes a hierarchy of generic models which describe transaction oriented problems (Figure 3.7), as well as a set of generic tasks which describe human goal-oriented activity [Sut02]. Domain theory’s generalised task models bear certain resemblance to Minsky’s frames [Min74], Schank’s scripts [SAS77], and Sowa’s schemata [Sow84]. From an object oriented perspective, the Domain Theory proposes a set of models built upon object collaborations that transform initial states into a single goal state.

The generic models hierarchy, called Object System Model (OSM) hierarchy, starts with nine separate trees, called the OSM families:

- **Object Containment** - describes transaction based applications, e.g. sales order processing, loans, inventory management;

- **Object Allocation** - describes applications that create an object state change as precursor to another transaction, e.g. booking type applications;

- **Object Composition** - describes applications that create one or more composite objects from components, e.g. assemble, manufacture;

- **Object Decomposition** - describes applications that disassemble objects into components;

- **Object Logistics** - describes applications that move goods and messages between an initial and final location, e.g. transport, transfer;

- **Object Construction** - describes applications that produce new objects by modifying components and possibly adding new elements during the construction process, e.g. build, manufacture;
3.6. MODELLING CONCEPTS

![Diagram of OSM hierarchy showing subclasses of the Object Containment Model][Sut02]

- **Object Sensing** - describes applications that monitor physical conditions or movements of objects, e.g. display monitors, motion sensors;

- **Agent Control** - describes applications that command and control, e.g. airspace traffic control;

- **Object Simulation** - describes decision support and interactive modelling applications, e.g. multimedia presentations, simulations, virtual environments;

Similarly, generalised tasks are represented as a goal hierarchy, organised in 11 families:

- **Information Acquisition** - describes the elicitation and acquisition of information from people or documented sources;

- **Analysis-Modelling** - describes the organisation of facts into a coherent representation (modelling) and the inference of properties of the representation;

- **Validation-Testing** - describes the verification of a design or problem solution in order to test that it satisfies a set of criteria;
• **Progress Tracking** - describes the progression of a process or an object-agent through different states;

• **Planning-Scheduling** - describes the scheduling of some resource or event so that it occurs at a specific time or place;

• **Navigation** - describes the planning, monitoring, and execution of a movement towards a certain destination;

• **Diagnosis** - describes the analysis and repair of a problem;

• **Information Retrieval** - describes the selection of a search strategy and search terms, the actual search for information items, and finally the evaluation of results relevance;

• **Judgement-Decision Making** - describes the course of action, the acceptance or rejection of options, or the selection of one or more options from a set of possibilities.

• **Explanation-Advising** - describes the tasks associated with learning;

• **Matching** - describes the process of finding the goodness of fit between requirements and properties of objects that might satisfy those requirements;

Generalised Tasks serve as models for defining concrete examples. For example, the generalised model for Diagnosis analyses and attempts to repair a problem, however instances of this model may be applied to medical diagnosis of an illness or to the technical diagnosis of a hardware problem.

OSMs with generic requirements attached are instruments of reusable knowledge that can be applied to elicit and define functional requirements in new applications. Furthermore, this tuple (OSM, generic requirements) also provides check-lists for identifying issues that require in-depth analysis (example in Figure 3.8).

### 3.6.4 Conceptual Graph

Conceptual graphs (CGs) are knowledge representation mechanisms informed by linguistics, psychology, and philosophy, that describe the way in which the human brain makes sense of the world. CGs have the following properties:

• **Finite** - human brains and other artificial memories have finite storage capacities;
3.6. MODELLING CONCEPTS

Figure 3.8: Object Sensing OSM with associated generic requirements [Sut02]

Figure 3.9: The sentence “A man reads a book” represented through Sowa’s conceptual graph.

- Connected - two parts that are not connected would become two CGs;

- Bipartite - there are two types of nodes (concepts and conceptual relations) and every arc links two different types of nodes;

There are two arguments that support conceptual graphs as representation mechanism for conceptual models [Sow84]. First, conceptual graphs support direct mapping to and from natural language. Second, conceptual graphs can easily be mapped to other logical notations, making them ideal as intermediate representations between natural languages and other formal symbolisms.

Figure 3.9 represents a conceptual graph using Sowa’s representation for an example sentence.
CHAPTER 3. FOUNDATIONAL WORK

3.7 Summary

This chapter has described the fundamental works used in this investigation. Initially, the examination of WordNet verb categories and Case Grammar have provided insight in the mechanics of the English language. Essentially, verbs organise sentences and can be used as starting points in the analysis of sentences. Through their predicate-argument structure, verbs create specific syntactic structures within which nouns with specific roles reside. The next chapter introduces and describes a type of such structures which semantically link verbs to nouns with certain roles.

Once these verb-centred structures have been created, there is a need for a method of applying them to NL. The purpose is the extraction of information useful for conceptual modelling. Chen pioneered an initial set of rules for directly translating NLRs into ER models [Che83]. These rules further strengthened the existence of a correspondence between NL and conceptual models. A decade later, Rolland and Proix propose a general linguistic approach for conceptual modelling, which incorporates some of Chen’s ideas and combines them with a semantic case approach [Fil67]. Similarly, in Chapter 5, a set of translation rules is presented. These rules are designed to map the semantic structures mentioned above to NLRs and extract useful information from the text. Additionally, these rules are designed to compose the simple semantic structures into more complex structures that provide a comprehensive view of the input NLRs.

Finally, the study of modelling concepts provided the necessary tools to represent the semantic models and their composition. Representational theory and the scenario meta-model provided useful insights for the design of a semantic meta-model. First, representational theory suggested the primitive ontological constructs that should exist in any functional conceptual meta-model. Second, the scenario meta-model provided a model of relationships between such concepts. Domain theory proposes an approach of linking generic conceptual models to real world tasks in an effort to analyse said tasks. Finally, conceptual graphs represent a method of visualising and structuring information. Consequently, the semantic models which are described in the next chapter represent an aggregate of these fundamental works.
Chapter 4

SOM Patterns

4.1 Introduction

In the previous chapter the foundational concepts underlying Semantic Object Models (SOMs) have been discussed in detail. The purpose of this chapter is to introduce the SOM patterns, their concepts, relationships, and definitions.

**Definition:** SOMs are requirements-centred primitive abstractions that connect a coherent set of modelling concepts with a specific WordNet verb category, with the purpose of extracting useful information from NLRs for conceptual modelling.

SOMs support information system modellers in the early stages of software development such as requirements analysis and modelling. SOM are based on nine WordNet verb categories (Section 3.2), therefore there are nine SOMs: Change SOM, Creation SOM, Motion SOM, Contact SOM, Possession SOM, Stative SOM, Cognition SOM, Perception SOM, and Communication SOM.

Each SOM has a conceptual representation in the form of a directed graph. Figure 4.1 displays the SOM notation. The graph contains two types of nodes, respectively boxes and circles. Boxes represent action elements (i.e. verbs), while circles represent things (i.e. entities). Actions and things are linked through directed arcs. These senses denote either the sense of an activity (e.g. someone performs and action on an object), the direction of resource allocation or travel (e.g. move an object from A to B), or the use of a resource (e.g. someone does something using a resource). The notation distinguishes between mandatory (enclosed in full circles) and optional (enclosed in dotted circles) concepts needed for comprehensive specification.
4.2 SOM Concepts

The proposed SOM approach is designed to guide the analysis process starting from NLRs containing descriptions of functional requirements (FRs). FRs describe what the system or system services should do and they should be complete. More specifically, they should include descriptions of all facilities required in the process. SOM patterns cater for completeness of FRs, through their structure, which is partly inspired from Fillmore’s case grammar [Fil67]. Because in the context of this research project FRs are represented in NL, the linguistic structure of SOMs inspired from case theory are ideal for identifying completeness of FRs. Current SOM concepts can be easily added.
4.2. SOM CONCEPTS

or modified to fit the scope of the models, that is to completely capture FRs.

The SOM concepts are divided into two groups: Thing and Action. Figure 4.2 summarizes SOM concepts and their groupings. Thing is a root concept that represents a set of entity concepts. Things are described by a set of properties. Thing concepts correspond to noun phrases in English language. They are further divided into Key Object, Agent, Container, Object, Instrument, and Material. These concepts are defined as follows:

**Key Object** is a Thing to be manipulated by some agent who performs an action and can be shared between agents. A key object has at least two states, initial and final. Key objects represent conceptual entities, e.g. purchase orders. This concept is mandatory in SOM.

**State** changes in key objects are caused by actions. In SOM, each key object will have at least two states, an initial state which represents the initial condition of the object and a final state. The final state of a key object is called the **goal** state.

**Agent** is a Thing that performs some action on key objects and causes key objects to change their states. The Agent concept is used to represent a person, a group of people (e.g. salesman, banker), or a machine (e.g. ATM, Order Management System). One agent can act upon another agent through a key object. Agent can be a **Source** or a **Destination Agent**.

**Container** is a Thing that represents storage for one or more key objects. Examples of containers are databases, shops, classrooms, and homes. When a key object is held in a container, the object is said to be a member of the container. The relationship between a key object and a container is the membership relation. Container can be a **Source** or a **Destination Container**.

**Instrument** is a Thing that represents a tool used by agents to perform some action on a key object. For example, a currency converter is an instrument for a human agent during money exchange.

**Material** is a Thing used to compose key objects. For example, aluminium alloy is a material for making a vehicle.

**Object** is a Thing that represents some resource consumed during an action. For example, each banking transaction will involve the update of the account concerned and the amount of money withdrawn or deposited will be represented as an object. The account is a key object.

The behaviour of actions is determined by a set of constraints. Action concepts correspond to verb phrases in English language. Action is a root concept that represents
actions performed by one or more agents on a key object, possibly involving additional things. Actions are classified into Action Category, which is further divided into the nine SOMs defined by the WordNet verb categories (Chapter 3): Change, Creation, Motion, Contact, Possession, Stative, Cognition, Perception, and Communication.

Figure 4.3 depicts the previously defined concepts and their relationships.

4.3 Change SOM

Change SOM is based on the Change verb category. It specifies the possible syntactic and semantic structures of the sentences associated with verbs of change. Change SOM is associated with at least one Agent, one Key Object (KO), zero or more Objects (OBJ), and zero or more Instruments (INST). Figure 4.4 shows Change SOM represented as a conceptual graph. Change SOM is used to describe the situation where an agent performs a change action that will affect the key object and cause it to change its state.
4.4 CREATION SOM

In active voice sentences, an agent represents the subject of the sentence. A key object represents the direct object of the transitive verb. In passive voice sentences, an agent represents the complement of a passive verb. A key object represents the syntactic subject of the sentence. In both active and passive voice sentences, objects and instruments represent verb prepositional modifiers. The agent may use an instrument or an object to modify the state of the key object.

Change SOM describes a wide variety of sentence structures that have the meaning of change. For example:

a) The layout \( \text{KO} \) was \( \text{CHANGE} \).

b) The developer \( \text{AGENT} \) changed \( \text{CHANGE} \) the layout \( \text{KO} \).

c) The customer \( \text{AGENT} \) changed \( \text{CHANGE} \) the yearly subscription for a monthly subscription \( \text{KO} \) \( \text{OBJ} \).

d) The researcher \( \text{AGENT} \) edited \( \text{CHANGE} \) the paper with a computer \( \text{KO} \) \( \text{INST} \).

e) The customer \( \text{AGENT} \) changed \( \text{CHANGE} \) the yearly subscription for a monthly subscription with \( \text{INST} \) a request form \( \text{KO} \) \( \text{OBJ} \).

f) The loan \( \text{KO} \) has been extended \( \text{CHANGE} \) by the librarian \( \text{AGENT} \).

4.4 Creation SOM

Creation SOM is based on the Creation verb category. It specifies the possible syntactic and semantic structures of sentences associated with verbs of creation. Creation SOM is associated with at least one Agent, one Key Object, and zero or more Materials (MAT). Figure 4.5 shows the Creation SOM represented as a conceptual graph.
CHAPTER 4. SOM PATTERNS

Creation SOM is used to describe the situation where an agent performs a creation action that results in the creation of the key object.

In active voice sentences, an agent represents the subject of the sentence. A key object represents the direct object of the transitive verb. In passive voice sentences, an agent represents the complement of a passive verb. A key object represents the syntactic subject of the sentence. In both active and passive voice sentences, materials represent verb prepositional modifiers. The agent may use some material to create the key object.

Creation SOM describes a wide variety of sentence structures that have the meaning of creation. For example:

a) Vehicles are assembled.

b) The robot assembles vehicles.

c) The robot assembles vehicles from prefabricated components.

d) The house has been built from timber by the workers.

4.5 Motion SOM

Motion SOM is based on the Motion verb category. It specifies the possible syntactic and semantic structures of sentences associated with verbs of motion. Motion SOM is associated with at least one Agent, one Key Object, zero or one Source Container (SRC CONT), and zero or one Destination Container (DST CONT). Figure 4.6 shows Motion SOM represented as a conceptual graph.
Motion SOM is used to describe the situation where an agent performs a motion action that will affect the key object and cause it to change its state.

In active voice sentences, an agent represents the subject of the sentence. A key object represents the direct object of the transitive verb. In passive voice sentences, an agent represents the complement of a passive verb. A key object represents the syntactic subject of the sentence. In both active and passive voice sentences, source and destination containers represent prepositional modifiers. The agent may move the key object from a source container to a destination container.

Motion SOM describes a wide variety of sentence structures that have the meaning of motion. For example:

a) Goods are moved.  
   KO  MOTION

b) The library moved the books.  
   AGENT MOTION  KO

c) The library moved the books from the north campus to the south campus.  
   AGENT MOTION  KO  SRC CONT  DST CONT

d) The book bar code is entered manually in the scanner by the customer.  
   KO  MOTION  DST CONT  AGENT

### 4.6 Contact SOM

Contact SOM is based on the Contact verb category. It specifies the possible syntactic and semantic structures of sentences associated with verbs of contact. Contact SOM is associated with at least one Agent, one Key Object, zero or one Container (CONT), zero or more instruments, and zero or more objects. Figure 4.7 shows Contact SOM represented as a conceptual graph. Contact SOM is used to describe the situation where an agent performs a contact action that will affect the key object and cause it to change its state.

In active voice sentences, an agent represents the subject of the sentence. A key object represents the direct object of the transitive verb. In passive voice sentences, an
agent represents the complement of a passive verb. A key object represents the syntactic subject of the sentence. In both active and passive voice sentences, containers, instruments, and objects represent prepositional modifiers. The agent may contact the key object placed within a container, or in the vicinity of an object, using an instrument.

Contact SOM describes a wide variety of sentence structures that have the meaning of contact. For example:

a) The valve is shut off.
   
   KO CONTACT

b) The engineer shuts off the valve.
   
   AGENT CONTACT KO

c) The engineer shuts off the valve through the console.
   
   AGENT CONTACT KO INST

d) The worker places the item inside the package.
   
   AGENT CONTACT KO CONT

e) The item is put on the conveyor belt by the customer.
   
   KO CONTACT OBJ AGENT

**4.7 Possession SOM**

Possession SOM is based on the Possession verb category. It specifies the possible syntactic and semantic structures of sentences associated with verbs of possession. Possession SOM is associated with at least one **Source Agent (SRC AGENT)**, zero or more **Destination Agents (DST AGENT)**, and one **Key Object**. Figure 4.8 shows Possession SOM represented as a conceptual graph. Possession SOM is used to describe this situation where an agent performs a possession action that will affect the key object and cause it to change its ownership either temporarily or permanently.

There are three subcategories of verbs of possession: destination possession (e.g. buying, getting, importing), source possession (e.g. selling, offering, exporting), and static possession (e.g. owning, having, possessing). In active voice sentences with destination possession actions, the destination agent represents the subject of the sentence...
4.8. Stative SOM

Stative SOM is based on the Stative verb category. It specifies the possible syntactic and semantic structures of sentences associated with stative verbs. Stative SOM is associated with at least one Thing, one Key Object, and zero or more Objects. Figure 4.9 shows the Stative SOM represented as a conceptual graph. Stative SOM is used to describe the situation where there is a static ownership relationship between thing and key object. The difference between Stative SOM and Static Possession SOM is
that the Stative Thing cannot semantically represent an Agent. Moreover, every other SOM that has an Agent that does not semantically correspond to the definition of agent (Section 4.2) is transformed into a Stative SOM.

In active voice sentences, a thing represents the subject of the sentence. A key object represents the direct object of the transitive verb. In passive voice sentences, a thing represents the complement of a passive verb. A key object represents the syntactic subject of the sentence. In both active and passive voice sentences, objects represent prepositional modifiers. An object may exist that offers more information on the static relationship between agent and key object.

Stative SOM describes a wide variety of sentence structures that have the meaning of stative relationship. For example:

a) A membership card shows a unique member number.

b) Along with the membership number, other details on a customer must be kept by the membership card such as a name, address, and date of birth.

4.9 Cognition SOM

Cognition SOM is based on the Cognition verb category. It specifies the possible syntactic and semantic structures of sentences associated with verbs of cognition. Cognition SOM is associated with at least one Agent, one Key Object, zero or one Container, and zero or more Objects. Figure 4.10 show Cognition SOM represented as a conceptual graph. Cognition SOM is used to describe the situation where an agent performs a cognition action that will affect the key object and
4.10. PERCEPTION SOM

cause it to change its state.

In active voice sentences, an agent represents the subject of the sentence. A key object represents the direct object of the transitive verb. In passive voice sentences, an agent represents the complement of a passive verb. A key object represents the syntactic subject of the sentence. In both active and passive voice sentences, containers and objects represent prepositional modifiers. The agent may perform cognition on the key object placed within a container, or change its state using an object.

Cognition SOM describes a wide variety of sentence structures that have the meaning of cognition. For example,

a) Textbooks are read .
KO COGNITION

b) Students read textbooks.
AGENT COGNITION KO

c) Students read textbooks from the curriculum.
AGENT COGNITION KO CONT

d) The system adds the price to the total.
AGENT COGNITION KO OBJ

e) Each customer is known as a member by the library.
KO COGNITION OBJ AGENT

4.10  Perception SOM

Perception SOM is based on the Perception verb category. It specifies the possible syntactic and semantic structures of sentences associated with verbs of perception. Perception SOM is associated with at least one Agent, one Key Object, and zero or more Instruments. Figure 4.11 shows Perception SOM represented as a conceptual graph. Perception SOM is used to describe the situation where an agent performs a perception action that will affect the key object and cause it to change its state.
In active voice sentences, an agent represents the subject of the sentence. A key object represents the direct object of the transitive verb. In passive voice sentences, an agent represents the complement of a passive verb. A key object represents the syntactic subject of the sentence. In both active and passive voice sentences, instruments represent prepositional modifiers. The agent may use some instrument to perform perception on the key object.

Perception SOM describes a wide variety of sentence structures that have the meaning of perception. For example:

a) The process is monitored.
   KO PERCEPTION

b) The scientist observes the cells.
   AGENT PERCEPTION KO

c) The scientist observes the cells through a microscope.
   AGENT PERCEPTION KO INST
   – (Agent:) scientist, (Perception Action:) observes, (Key Object:) cells, (Instrument:) microscope;

d) The display is observed by the engineer through a viewfinder.
   KO PERCEPTION AGENT INST

4.11 Communication SOM

Communication SOM is based on the Communication verb category. It specifies the possible syntactic and semantic structures of sentences associated with verbs of communication. Communication SOM is associated with at least one Source Agent, at least one Destination Agent, and one Key Object. Figure 4.12 shows Communication SOM represented as a conceptual graph. Communication SOM is used to describe the situation where at least two agents perform a communication action that will affect the key object and cause it to change its state.
Communication SOM describes a wide variety of sentence structures that have the meaning of communication. There are two subcategories of verbs of communication: direct communication, in which an agent is present right after the communication verb (e.g. ask, query, beg, inform, update someone) and indirect communication, in which the object of communication is present right after the communication verb (e.g. issue, request, approve, submit something). In active voice sentences with direct communication the source agent represents the subject of the sentence, the destination agent represents the direct object of the transitive verb, and the key object represents a prepositional modifier (Example a). In passive voice sentences with direct communication the source agent represents the complement of a passive verb, the destination agent represents the subject of the sentence, and the key object represents the direct object of the transitive verb (Example b).

a) Students query the database for books.
   SRC AGENT COMMUNICATION DST AGENT KO

b) Librarians are asked questions by students.
   DST AGENT COMMUNICATION KO SRC AGENT

In active voice sentences with indirect communication the source agent represents the subject of the sentence, the destination agent represents a prepositional modifier (Example c). In passive voice sentences with indirect communication the source agent represents the complement of a passive verb, the destination agent represents the subject of the sentence (Example d). In active voice sentences with indirect communication, the key object represents the direct object of the transitive verb (Example e). In passive voice sentences with indirect communication, the key object represents the syntactic subject of the sentence (Example f).

c) A library issues loan items to its members.
   SRC AGENT COMMUNICATION KO DST AGENT

d) Each customer is issued a membership card by the library.
   DST AGENT COMMUNICATION KO SRC AGENT

e) Students submit project reports before deadlines.
   SRC AGENT COMMUNICATION KO

f) Request documents are submitted to the claims department by case officers.
   KO COMMUNICATION DST AGENT SRC AGENT
4.12 Summary

This chapter has described the nine primitive semantic object models upon which the SOM approach has been developed and their concepts. Each SOM is defined following a template containing a graphical representation of the SOM, a description of the structure and its goal, and some sentence examples for both active and passive voices.
Chapter 5

SOM Translation Rules and Application

5.1 Introduction

The purpose of this chapter is to present a set of rules for translating NLRs into SOM instances (SOMi). Figure 5.1 shows an overview of the SOM translation rule sets. The rule sets are made of 13 syntactic, ten semantic, four composition rules, and eight NL generation rules. Whereas syntactic rules are similar to those proposed by Chen [Che83] realating to identifying structures, semantic rules identify semantic agents, behaviour, and constraints, composition rules guide the transformation of NLRs into conceptual models, and NL generation rules direct the transformation of the information captured through SOMs into NL text for user validation.

Figure 5.1: SOM Translation Rule Sets
5.2 Syntactic Rules

There are three sets of syntactic rules, one for active voice sentences, one for passive voice sentences, and one for both active and passive sentences. Syntactic rules take advantage of relationships between words in English sentences to identify concepts such as subjects, complements, or direct objects, which are useful for the SOM approach. These rules are described in the following sections.

5.2.1 Active Voice

There are six rules for active voice sentences, which are described as follows.

Rule 1 - Identifying agents. If the syntactic subject of a clause is a noun phrase, then it is considered as a agent candidate.

Rule 2 - Determining key objects. Except for direct communication verbs, a direct object of a verb phrase is assigned the role of key object.

Example Statement: The salesman creates the monthly report.
Translation: “Salesman” is the subject of the sentence and also an agent candidate (Rule 1). “Monthly report” is the direct object of the transitive verb “creates” and is a key object (Rule 2).

Rule 3 - Identifying agents. An external subject of an open clausal complement is considered as an agent candidate.

Example Statement: The analyst wants to create the document.
Translation: “Analyst” is the external subject of the example statement, which is considered as the agent candidate who creates the document.

Rule 4 - Identifying destination agents. For direct communication verbs, a direct object of a verb phrase is considered as a destination agent candidate.

Rule 5 - Determining key objects. For direct communication verbs, a noun phrase introduced by the “of”, “for”, “about”, or “with” prepositions is assigned the role of key object.

Example Statement: The nurse informs the doctor of the patients.
Translation: “Patients” are assigned the role of key object (Rule 5) and “doctor” is considered as a destination agent candidate (Rule 4).

Rule 6 - Identifying destination agents. For indirect communication verbs a noun phrase introduced by the “to” prepositions is considered as a destination agent candidate.

Example Statement: Library issues items to customers.
5.2. SYNTACTIC RULES

Translation: “Customers” are destination agent candidates.

5.2.2 Passive Voice

There are four rules for passive voice sentences, which are described as follows.

Rule 7 - Identifying agents. The complement of a passive verb that is introduced through the preposition “by” is an agent candidate.

Rule 8 - Determining key objects. If the syntactic subject of a passive clause is a passive nominal subject, it is assigned the role of key object.

Example Statement: The report is submitted by the officer.

Translation: “Officer” is an agent candidate (Rule 7), and “report” is the key object (Rule 8).

Rule 9 - Identifying destination agents. For indirect communication verbs, if the syntactic subject of a passive clause is a passive nominal subject, it is considered as a destination agent candidate.

Rule 10 - Determining key objects. For indirect communication verbs, a direct object of a phrase is assigned the role of key object.

Example Statement: Each customer is issued a card.

Translation: “customer” is a destination agent candidate (Rule 9), and “card” is the key object (Rule 10).

5.2.3 Rules for Active and Passive Voices

There are three additional rules for both active and passive voice sentences, which are described as follows.

Rule 11 - Identifying source agents. For possession and communication verbs, a prepositional modifier of the verb introduced by the “from” preposition is considered as a source agent candidate.

Example Statement: Michael bought the car from Jordan.

Translation: “Jordan”, which is a proper noun that is part of a prepositional phrase, is a source agent candidate.

Rule 12 - Identifying destination agents. For possession and communication verbs, a prepositional modifier of the verb introduced by the “to” preposition is considered as a destination agent candidate.

Example Statement: Michael sold the car to Jordan.
**Translation:** “Jordan”, which is a proper noun that is part of a prepositional phrase, is a destination agent candidate.

**Rule 13 - Determining instruments, objects, containers, and materials.** Other prepositional modifiers, such as “with”, “near”, “under”, “on”, “in”, “inside”, are considered as Instrument, Object, Container, or Material. Their specific roles will be decided according to specific SOMs.

*Example Statement:* I placed my laptop into my suitcase.

*Translation:* “Suitcase”, which is introduced by the prepositional modifier “into” is a container.

### 5.3 Semantic Rules

Semantic rules in the SOM approach carry out three roles: (1) to determine if a syntactically identified concept can represent an agent, (2) to determine inter-SOM behaviour, and (3) to find constraints in the NLRs.

This section presents the set of semantic rules.

**Rule 14 - Determining agents.** If a candidate agent denotes a person, a group of people, or a machine, then it is classified as an agent, else it is classified as a non-agent.

*Example Statement:* The salesman creates the monthly report.

*Translation:* “Salesman” is an agent candidate that denotes a person, thus we conclude that it is an agent.

*Example Statement:* Numbers denote sections of a library.

*Translation:* “Numbers” is also a candidate agent according to Rule 1, however it does not denote a person, a group of people, or a machine, thus it is classified as a non-agent.

**Rule 15 - Determining sequence behaviour.** If the requirement sentence has the form “X \(\text{Adv}_{T_1} \ Y\)”, where \(\text{Adv}_{T_1}\) = \{followed by, and then, then\} and X and Y are verbs, we determine that action denoted by verb X occurs before action denoted by verb Y.

*Example Statement:* The card is entered and then the pin number is requested.

*Translation:* After the “enter” activity occurs, the “request” activity follows.

**Rule 16 - Determining sequence behaviour.** If the requirement sentence has the form “Y \(\text{Adv}_{T_2} \ X\)”, where \(\text{Adv}_{T_2}\) = \{after, when, if, once, following\} and X and Y are verbs, we determine that action denoted by verb X occurs before action denoted by verb Y.
Example Statement: The pin number is requested after the card details have been read.

Translation: First, the “read” activity occurs, then the “request” activity follows.

Rule 17 - Determining parallel split behaviour. If the requirement sentence has the form “Adv_{T_2} W X, Y, and Z”, where Adv_{T_2}={after, when, if, once, following} and W, X, Y, and Z are verbs, we determine that activities denoted by verbs X, Y, and Z take place after the action denoted by verb W occurs.

Example Statement: Once the identity is confirmed, the funds are transferred and the balance updated.

Translation: Confirmation activity occurs first, followed by the transfer of funds and update of the account balance in parallel.

Rule 18 - Determining synchronization behaviour. If the requirement sentence has the form “Adv_{T_2} X, Y, and Z, W”, where Adv_{T_2}={after, when, if, once, following} and W, X, Y, and Z are verbs, we determine that activities denoted by verbs X, Y, and Z take place before the action denoted by verb W can start.

Example Statement: Once the funds are transferred and the balance updated, the transaction must be recorded.

Translation: The transfer and update activities must finish before the transaction can be recorded.

Rule 19 - Determining synchronization behaviour. If the requirement sentence has the form “W Adv_{T_2} X, Y, and Z”, where Adv_{T_2}={after, when, if, once, following} and W, X, Y, and Z are verbs, we determine that activities denoted by verbs X, Y, and Z take place before the action denoted by verb W can start.

Example Statement: The wound has to be dressed after it has been disinfected and cleaned.

Translation: The disinfection and cleaning of the wound must finish before the wound can be dressed.

Rule 20 - Determining XOR behaviour. If the requirement sentence has the form “Adv_{T_2} X either Y, or Z”, where Adv_{T_2}={after, when, if, once, following} and X, Y, and Z are verbs, we determine that after activity denoted by verb X occurs, the thread of control is immediately passed to precisely one of the outgoing activities denoted by verbs Y, and Z.

Example Statement: Following the mandatory service, the customer can either pick the car up by himself or have it delivered by one of our staff.

Translation: After the servicing activity occurs, the client can choose to pick the
car by himself or have it delivered.

**Rule 21 - Determining constraints.** If a clause is introduced by a subordinating conjunction (e.g. if, before, unless, when, where etc.), then its position in the text is recorded and the clause itself is recorded as a constraint.

*Example Statement:* If the account balance is low, a warning message is displayed on the screen and the card is ejected.

*Translation:* A warning message is displayed on the screen and the card is ejected in case the condition “if the account balance is low” is true.

### 5.4 Composition Rules

There are four composition rules in the SOM approach, which carry out two main roles: (1) compose SOMi relating to an NLR into a comprehensive Semantic Object Network (SON) in order to offer a comprehensive view of the information extracted from the input NLR; (2) attach behaviour and constraints to SOMi. The composition rules are partly inspired from Sowa’s generalization and specialization rules [Sow84].

**Rule 22 - Creating composition keys.** All SOMi elements are lemmatised and their lexemes are used as composition keys.

**Rule 23 - Composing SOMi using composition keys.** If an instance of one concept type (e.g. Thing or Action) appears in many SOMi, then all the information related to that concept type instance is gathered under one key.

For example, “patient” can be an agent in one SOMi and a key object in another SOMi. Therefore, after composition there will be only one “patient” instance.

**Rule 24 - Attaching behaviour to verbs.** If two or more verbs are positioned in a valid behaviour pattern (Rules 15 - 20) then the resulting pattern behaviour is attached to those verbs.

**Rule 25 - Attaching constraints to verbs.** If a recorded constraint contains a verb, then the constraint will be attached to that verb.

*Example Statement:* If the account balance is low, a warning message is displayed on the screen to the user and the card is ejected.

*Analysis:* The SOMi elements have been brought to a canonical form (*displayed* - display, *ejected* - eject) (Rule 22). The clause “if the account balance is low” is recorded as a constraint (Rule 21) and is linked to the “ATM displays warning message” and “ATM ejects card” SOMi (Rule 25). The parallel split behaviour rule is
triggered. Thus, “ATM displays warning message” and “ATM ejects card” will be executed in parallel (Rule 17, Rule 24). The SON represents this behaviour by linking the two verbs with a node such as in Figure 5.2.

### 5.5 NL Generation Rules

Currently, there are eight NL generation rules in the SOM approach. Their role is to generate NL sentences from the information captured by the SOMs so that users can validate the information captured. The first four rules generate simple requirements sentences, whereas the last four rules compose the simple requirements sentences into more complex sentences describing behaviour.

**Rule 26 - Generating NL requirements with one agent type.** When there is only one type of agent involved, the sentence is constructed as **Agent-Action-Key Object-other Things**, with cardinalities attached to all elements except verbs.

\[
S_i \rightarrow (\sum c_{N_{Agent}}) V_{Verb} N_{KeyObject} \sum prep_i c_{N_{Other Things}}
\]

**Example Statement:** The ATM updates the balance with the amount withdrawn.

---

Figure 5.2: Example of SON representation of behaviour and constraints

Figure 5.3: SOMi of the example statement
Translation: Figure 5.3 shows the SOMi for the example statement. The NL requirement generated from the SOMi is: “One ATM updates one balance with one amount”.

**Rule 27 - Generating NL requirements for direct communication.** In the case of direct communication, the paraphrased sentence is constructed as Source Agent-Action-Destination Agent-Key Object, with cardinalities attached to all elements except verbs.

\[
S_i \rightarrow (\sum c_{N_{Agent_{src_i}}})Verb_i(\sum c_{N_{Agent_{dst_i}}})prep_i c_{N_{keyObject_i}}
\]

Example Statement: The web-client queries the database for updated prices.

Translation: Figure 5.4 shows the SOMi for the example statement. The NL requirement generated from the SOMi is: “One web-client queries one database for many prices”.

**Rule 28 - Generating NL requirements for destination possession or indirect communication.** In the case of destination possession or indirect communication, the paraphrased sentence is constructed as Destination Agent-Action-Key Object-Source Agent, with cardinalities attached to all elements except verbs.

\[
S_i \rightarrow (\sum c_{N_{Agent_{dst_i}}})Verb_i c_{N_{keyObject_i}} \sum prep_i c_{N_{Agent_{src_i}}}
\]

Example Statement: The library issues loan items to students daily.

Translation: Figure 5.5 shows the SOMi for the example statement. The NL requirement generated from the SOMi is: “One library issues many loan items to many students”.

Figure 5.4: SOMi of the example statement

Figure 5.5: SOMi of the example statement
Rule 29 - Generating NL requirements for source possession. In the case of source possession, the paraphrased sentence is constructed as *Source Agent-Action-Key Object-Destination Agent*, with cardinalities attached to all elements except verbs.

\[
S_i \rightarrow (\sum c_{N_{\text{Agent}_{srci}}})\text{Verb}_i N_{\text{keyObject}} \sum \text{prep}_i c_{N_{\text{Agent}_{dsti}}}
\]

Example Statement: IBM sold several hundred servers to the German government last year.

Translation: Figure 5.5 shows the SOMi for the example statement. The NL requirement generated from the SOMi is: “One IBM sold many servers to one German government”.

Rule 30 - Generating NL requirements with sequence behaviour. In the case of at least two SOMs being captured within a sequence behaviour pattern, the paraphrased sentences is constructed as follows:

\[
S \rightarrow \left( \sum_{i=1}^{n-1} SOM_i \right) \text{“and then” } SOM_n
\]

Example Statement: A warning message is displayed on the screen to the user and then the card is ejected.

Translation: Assuming that the SOMi are complete (i.e. have no empty nodes), the NL requirement generated from the SON is: “One ATM displays one warning message to one user and then one ATM ejects one card.”

Rule 31 - Generating NL requirements with parallel behaviour.

\[
S \rightarrow \text{“After” } SOM_1, \left( \sum_{i=2}^{n-1} SOM_i \right) \text{“and” } SOM_n
\]

Example Statement: Once the bank confirms the identity of the customer, the funds are transferred in the account and the balance updated.

Translation: Assuming that the SOMi are complete (i.e. have no empty nodes), the NL requirement generated from the SON is: “After one bank confirms one identity,
one bank transfers many funds to one account and one system updates one balance.”

Rule 32 - Generating NL requirements with synchronization behaviour.

\[ S \rightarrow \text{"After" } \left( \sum_{i=1}^{n-2} \text{SOM}_i \right) \text{ "and" } \text{SOM}_{n-1}, \text{ "then" } \text{SOM}_n \]

Example Statement: The wound has to be dressed after it has been disinfected and cleaned.

Translation: Assuming that the SOMi are complete (i.e. have no empty nodes), the NL requirement generated from the SON is: “After one nurse disinfects one wound and one nurse cleans one wound, then one nurse dresses one wound.”

Rule 33 - Generating NL requirements with XOR behaviour.

\[ S \rightarrow \text{"Either" } \left( \sum_{i=1}^{n-1} \text{SOM}_i \right) \text{ "or" } \text{SOM}_n \]

Example Statement: Following the mandatory service, the customer can either pick the car up by himself or have it delivered by one of our staff.

Translation: Assuming that the SOMi are complete (i.e. have no empty nodes), the NL requirement generated from the SON is: “Either one customer picks up one car from one service, or one staff delivers one car to one customer.”

5.6 Rule Application Steps

This section follows on from the previous sections, which examined the detailed SOM definitions (Chapter 4) and the SOM translation rules. The purpose of this section is to describe the approach used to derive requirements models using the SOMs and translation rules. Figure 5.7 shows the SOM approach as a series of process steps and outputs. The first step is to determine the constraints from the input NLR document. The second step is to determine the behaviour between actions described in the input NLR. The third step determines SOM types and instantiates the SOM patterns. The fourth step combines the previously determined information into a SON model. Finally, the fifth step generates a document that contains NL generated requirements (NLG) ready for user validation. The SOM approach is illustrated using a running example from a library domain [Cal94]:
5.6. **RULE APPLICATION STEPS**

A library issues loan items to customers. Each customer is known as a member and is issued a membership card that shows a unique member number. Along with the membership number, other details on a customer must be kept such as a name, address, and date of birth. The library is made up of a number of subject sections. Each section is denoted by a classification mark. A loan item is uniquely identified by a bar code. There are two types of loan items, language tapes, and books. A language tape has a title language (e.g. French), and level (e.g. beginner). A book has a title, and author(s). A customer may borrow up to a maximum of 8 items. An item can be borrowed, reserved or renewed to extend a current loan. When an item is issued the customer’s membership number is scanned via a bar code reader or entered manually. If the membership is still valid and the number of items on loan less than 8, the book bar code is read, either via the bar code reader or entered manually. If the item can be issued (e.g. not reserved) the item is stamped and then issued. The library must support the facility for an item to be searched and for a daily update of records.

### 5.6.1 Step 1: Determine Constraints

Using Rule 21 (Section 5.3) the text is scanned in search of constraint patterns. If any matches are found, the constraint and the verbs involved are recorded. For example:

- **When an item is issued** the customer’s membership number is scanned via a bar code reader or entered manually - a membership number is scanned in the context of an item being issued;

- **If the membership is still valid and the number of items on loan less than 8**, the book bar code is read, either via the bar code reader or entered manually - the book’s bar code is read only if both conditions are true;
• If the item can be issued (e.g. not reserved) the item is stamped and then issued - finally, during the issuing process, if the membership is valid and the item is available, the process can successfully end.

5.6.2 Step 2: Determine Behaviour

First, verbs are identified in the text. Second, using Rules 15 to 20 (Section 5.3) the text is scanned in search of behaviour patterns. If any matches are found, the information regarding the type of behaviour and the verbs involved is recorded. For example:

• Each customer is known as a member and is issued a membership card that shows a unique member number - parallel behaviour linked to the verbs “know” and “issue” in which customers are issued membership cards at the same time as they register as members;

• An item can be borrowed, reserved or renewed to extend a current loan - XOR behaviour linked to the verbs “borrow”, “reserve”, and “renew”, which captures the three actions that can be performed on an item at one time;

• When an item is issued the customer’s membership number is scanned via a bar code reader or entered manually - XOR behaviour linked to the verbs “scan” and “enter”, which describes the two ways in which a membership number can be input in the system;

• If the item can be issued (e.g. not reserved) the item is stamped and then issued - sequence behaviour linked to the verbs “stamp” and “issue” describing the order in which the issuing process occurs.

5.6.3 Step 3: Determine SOM type and Instantiate SOM

This step of the process is composed of two parts. First, for each verb in the input NLR, its first meaning is used to select the relevant SOM type. The meaning of the verbs are extracted from WordNet’s database. For example:

• “…library issues loan items …” - the first sense of the verb “issue” is communication, thus, the Communication SOM is selected

• “…customer is known as a member …” - the first sense of the verb “know” is cognition, thus, the Cognition SOM is selected
5.6. RULE APPLICATION STEPS

- “... that shows a unique member number ...” - the first sense of the verb “show” is perception, thus, the Perception SOM is selected
- “... must be kept such as ...” - the first sense of the verb “keep” is stative, thus, the Stative SOM is selected

Second, after the SOM type has been established, Rules 1 through 14 are used to instantiate the SOM with elements extracted from the input NLR. For example, for the verbs identified above:

- *A library issues loan items to customers* - this is an active voice sentence represented by a Communication SOM, which triggers Rules 1 and 14 which determine the source agent as “library”, Rule 2 which determines the key object “loan items”, and Rules 6 and 14 which determine the destination agent “customers”.

- *Each customer is known as a member and is issued a membership card that shows a unique member number* - this phrase contains three sentences. The first is a passive voice sentence represented by a Cognition SOM, which triggers Rule 8 which determines the key object “customer” and the question “Who is the agent that knows the customer?”. The second is a passive voice sentence represented by a Communication SOM, which triggers Rules 9 and 14 which determine the destination agent as “customer”, Rule 10 which determines the key object “membership card”, and the question “Who is the agent that issues membership cards to customers?”. The third is an active voice sentence represented by a Perception SOM, which triggers Rules 1 and 14 which change the SOM type to Stative (Chapter 4) and determine a thing “membership card”, and Rule 2 determines the key object “member number”;

- *Along with the membership number, other details on a customer must be kept such as a name, address, and date of birth* - this is a passive voice sentence represented by a Stative SOM, which triggers Rule 8 which determines the key object “details”, Rule 13 which determines the objects “name”, “address”, “date of birth”, “membership number”, and the question “What thing keeps details such as name, address, date of birth, and membership number?”.

NLR input documents are seldom complete. Therefore, SOMi may not be completely filled after passing over the text. Empty SOMi nodes trigger elicitation questions that need to be addressed to the domain expert in order to extract the necessary information to complete the SOMi.
5.6.4 Step 4: Compose SOMi

In this step individual SOMi are composed into a SON using Rules 22-25. In this process two resources are involved. First, a set of SOMi, which is traversed in a linear fashion. Initially, the SON is composed of the first SOMi. Each SOMi thereafter is added to the SON using Rules 22 and 23. Once the SON is completed, the second resource is involved. This resource contains the behaviour and constraints, which are again traversed linearly and added to the SON using rules 24 and 25. Figure 5.8 shows the complete SON for the Library example.

Figure 5.8: SON for the Library Callan example. Double lined circles represent captured behaviour patterns that link specific SOMi; hexagons represent identification keys mapped to captured constraints that are linked to specific actions.

5.6.5 Step 5: Validate SON

SONs are conceptual models that capture domain specific information from input NLRs. Consequently, they have a dual role. First, SONs act as a communication tool between system developers and users during the process of identifying information requirements. Second, they are the starting point from which software design models
are developed. Errors made during this early phase of software development propagate to all the following phases, making them expensive to correct [Dav93, LBSL87]. Therefore, it is crucial to validate that (1) the models are consistent and (2) that they correspond to the informal description provided by users.

First, to ensure that the models are consistent, the SON is traversed following the arcs’ senses using a modified depth-first search method (Algorithm 1). The SON is traversed starting with the first node of the first SOMi, following the sense of the arcs in a depth-first search (DFS). The difference from the classic DFS algorithm stems from the shared resource property of key objects within the SOM approach. Key objects can link separate partitions such as systems or processes, without being explicitly passed between the respective partitions. They provide this link just by being accessed. For example, two different departments editing information in the same database table are considered to be linked. Once MDFS reaches a key object, all inbound arcs are considered accessible for traversal. Thus, in a system there are two ways of establishing connections: either through direct interaction (e.g. message sending, communication), or indirect interaction (e.g. shared resources). If there are no sub-graphs identified, the SON is considered to be consistent. Otherwise, it is considered that there is an inconsistency present (e.g. inconsistent use of terms such as synonyms or abbreviations, missing requirements), which if solved would connect the sub-graphs. Therefore, further elicitation questions regarding this issue are addressed to the user.

**Algorithm 1:** Modified depth-first search algorithm (MDFS) for determining SON completeness
Second, in order to ensure that the SON corresponds to the initial informal description, NL requirements are generated from the previously captured information as to facilitate their inspection by domain experts. To this extent, all the distinct SOMi that composed the SON are taken individually and their elements are composed into simple active voice sentences using Rules 26-29. Thereafter, NL requirements containing the dynamic behaviour are constructed using Rules 30-33. Figure 5.9 shows a SON fragment that contains behaviour between two actions. The simple NL sentences generated in this case would be: “One customer enters one code”, “One barcode reader reads one code”, and the sentence containing the dynamic behaviour would be: “Either one customer enters one code or one barcode reader reads one code”.

5.7 Summary

This chapter has started by describing the SOM translation rules. The purpose of these rules is to support the transformation of NLRs into conceptual models by taking advantage of the correspondence between English and SOM structures. These rules are divided into four sets: syntactic, semantic, composition, and NL generation rules. Syntactic rules deal with the identification of modelling concepts. Semantic rules identify semantic agents and dynamic aspects of requirements. Composition rules describe the guidelines for obtaining a comprehensive SON, which describes the input NLRs. Finally, NL generation rules guide the transformation of SOMi into NL text for user validation.

The final part of this chapter has described the five step process used to obtain and validate conceptual models using the SOM approach. Initially the NLR input is analysed for behaviour and constraint patterns. The results are recorded in order to
be inserted in the SON output. Next, all the verbs in the input NLR are analysed in order to select the appropriate SOM types based on the verbs’ most popular sense as identified by WordNet. After the SOM type has been successfully identified, the SOMi are obtained using the SOM translation rules. These rules map the elements present in the input NLR to conceptual modelling elements, present in the SOM structures. Next, the SOMi are composed into the SON to offer a comprehensive view of the input NLR for analysis. Finally, in order to validate the SON model, the SOMi are paraphrased into NL sentences.
Chapter 6

Tool Support for the SOM Approach

6.1 Introduction

A prototype tool based on the principles of the approach has been developed in order to validate the SOM approach [LZC13]. This chapter describes and discusses the software tool used in this investigation - Textual Requirements to Analysis Models (TRAM). This chapter is divided into four main sections, each corresponding to TRAM's software components. Each section is further divided in two parts. The first part discusses the design of the software components, while the second part describes the implementation details.

TRAM consists of four main software components (NLR Parser, SOM Constructor, SOMi Composer, NLR Generator) that carry out the rule application steps described in Section 5.6. Figure 6.2 shows the relationship between the application steps, software components, and documents.

TRAM is implemented in the Java programming language and based on the Eclipse

![Figure 6.1: Overview of TRAM software components (NLR Parser, SOM Constructor, SOMi Composer, NLR Generator) mapped over the SOM approach](image)
Figure 6.2: TRAM Architecture, showing its main software components (NLR Parser, SOM Constructor, SOMi Composer, NLR Generator), Control Flow(solid arrows) and Data Flow (dotted arrows)

Modelling Framework (EMF) tool suite. The TRAM project contains 18 packages, 133 classes, and over 37,000 lines of code. The overall architecture of TRAM is depicted in Figure 6.2, which shows its main components and documents, as well as their relationships (both control flow and dataflow). The process of TRAM consists of the following steps.

### 6.2 NLR Parser

#### 6.2.1 Component Design

The SOM approach takes as input unstructured, unrestricted natural language at any level of abstraction. Input documents can contain both functional and non-functional requirements, written in either active or passive voice, not necessarily in a consistent, disambiguated manner.

The parser performs five tasks: (1) identifying and assigning part-of-speech (POS) tags to the words in text (e.g., noun, verb, adjective, etc); (2) creating grammatical relations or type dependencies among elements in a sentence; (3) recording the position and assigning a unique identifier to each word in the text for traceability purposes; (4) identifying constraints introduced by subordinating conjunctions; (5) identifying behaviour patterns between SOMs.

NLR Parser transforms the input NLR into a common data structure enabling further computational processing. It accomplishes this by first assigning parts-of-speech
tags to words. In order to achieve this, the text is segmented into word and sentence units and each of these units are initially assigned POS-tags based on a lexicon and a set of rules. Next, the initial POS-tags are revised based on rule-driven contextual POS assignments, the probability of each potential sequence of tags is calculated, and the sequence with the highest probability is chosen.

After the identification and assignment of POS-tags, the second task is to represent sentence structures through typed dependencies. A dependency parse represents dependencies between individual words. Additionally, a typed dependency parse labels dependencies with grammatical relation labels such as subject or direct object. These grammatical relations provide information about predicate-argument structures which are not readily available from syntactic analysis [DMMM06].

During the initial POS tagging process, the input NLR is decomposed into a sequence of tokens, which correspond to the words in the text. The third task of the NLR parser is to mark the position of each token in the NLR and assign them IDs. This is the initial step that ensures requirements traceability from NLR input to conceptual models.

The fourth task is to identify constraints in the input NLR. Constraints in the SOM approach are defined as restrictions under which certain actions take place. For example, in the sentence “If the item can be issued, the item is stamped and then issued”, the actions of stamping and issuing occur under the restriction that the item can be issued. In order to carry out this task the NLR Parser uses Rule 21 (Section 5.3).

Finally, the fifth task involves identifying behaviour patterns between verbs. In order to carry out this task the NLR Parser uses Rules 15-20 (Section 5.3).

6.2.2 Implementation

NLR Parser is based on the Stanford parser [KM03] and used specifically for processing NLR. The Stanford parser is an unlexicalized natural language parser. TRAM uses version 3.2.0 of the parser, which requires Java 6 (JDK1.6) or later. The output of the NLR parser is saved as an XML file (Parsed NLR) containing SOM relevant information such as part of speech tags, word dependencies, identification information, and constraints. Figure 6.3 shows the structure of the Parsed NLR, where: dep type denotes the dependency type; id denotes the position of the word in the text, that is, its beginning and ending character offsets; pos represents the assigned part-of-speech tag; cardinality is a measure of the number of elements the word suggests (singular or plural); word denotes the actual word from text.
6.3 SOM Constructor

6.3.1 Component Design

SOM Constructor uses the output of the NLR Parser to generate a set of SOMs as intermediate models to bridge the gap between the input NLR and its conceptual model. SOM Constructor uses all the verbs from the input NLR to identify the suitable SOM patterns. Since each verb may have multiple senses, SOM Constructor will select the most commonly used sense from the WordNet database and map it onto a corresponding SOM pattern.

After each SOM pattern is identified, the SOM Constructor will look for its associated concepts from the typed dependency parse. For example, the sentence “A library issues loan items to customers” is a Communication SOM that is associated with the following concepts: a Communication action (issues), a Source Agent (library), a Key Object (loan items), and a Destination Agent (customers). These concepts are extracted from the typed dependency relations displayed in this statement: loan items are
the direct object of the verb *issues*, *library* is the nominal subject of the sentence, and *customers* is the prepositional object of the verb *issues*.

Fully automated instantiation may not completely fill all the SOM nodes due to the inherent problems of natural language such as ambiguities (e.g. use of co-references such as pronouns) and omissions (e.g. inherently missing concepts such as unspecified agents). However, in contrast to current related approaches, the SOM approach guides the user, indicating exactly what information is missing by highlighting empty SOM nodes and prompting the user to fill in the missing concepts. This process is called the Question and Answer activity (Q&A) and it is driven by the semantic information encoded within the SOM patterns.

### 6.3.2 Implementation

WordNet is an electronic lexical database for English whose design was inspired by psycholinguistic theories of human lexical memory [MBF+90]. This database organises English nouns, verbs, and adjectives into synonym sets, linked through semantic relations. WordNet is a hierarchy of concepts, where concepts that are more specific inherit information from more general concepts. Made possible by its electronic format, WordNet has become a major tool for natural language processing, and has motivated research in lexical semantics and ontology [Fel10]. TRAM uses version 3.1 of the WordNet dictionary for selecting most popular senses of verbs in order to select a

```xml
<?xml version="1.0" encoding="ASCII"?>
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  <hasRequirement id="1" name="issue item">
    <hasAction name="issues" id="10.16" pos="VBZ" cardinality="1">
      modifier="" alternative="" lemma="issue"/>
    </hasAction>
    <hasKeyObject name="items" id="22.27" pos="NNS" cardinality="*">
      modifier="" alternative="loan item" lemma="item"/>
    </hasKeyObject>
    <hasAgent name="customers" id="31.48" pos="NNS" cardinality="*">
      modifier="prep_to" alternatives="" lemma="customer"/>
    </hasAgent>
    <hasTarget name="library" id="2.9" pos="NN" cardinality="1">
      modifier="" alternative="library"/>
    </hasTarget>
  </hasRequirement>
  ...
  <hasBehaviour id="1" components="2 3 " type="P.5"/>
  <hasBehaviour id="4" components="21 22 " type="XOR"/>
  <hasBehaviour id="5" components="23 24 25 " type="SEQ"/>
  <hasConstraint id="9" components="19" value="If the item can be issued" nId="1763578647"/>
  <hasConstraint id="9" components="20" value="When an item is issued" nId="1056359207"/>
</som:FunctionalSpec>
```

Figure 6.4: XMI snippet of a SOMi
suitable SOM Pattern.

Each SOM pattern is modelled as a Java class. The SOM patterns are organized in a Strategy Pattern [GHJV93]. Moreover, the SOM translation rules are incorporated within each SOM class. The strategy pattern is used to select the appropriate translation rules associated with each SOM. Translation rules are used to generate SOMi, which are represented in XMI format, as shown in Figure 6.4.

After the application of the SOM translation rules the SOMi may not be complete. To perform elicitation of the missing instances, TRAM interacts with the user through the Q&A dialogue process. In order to provide the missing instances the user can select an instance from the drop-down menu or by inputting an instance manually.

In order to generate the SOMi XMI structure, the SOM meta-model is represented in ECORE, which is a standard representation within the EMF project. Figure 6.5 shows the SOM ECORE meta-model. EMF is a modelling framework and code generation facility for building tools and other applications based on a structured data model. From a model specification described in XMI, EMF provides tools and runtime support to produce a set of Java classes for the model, along with a set of adapter classes that enable viewing and command-based editing of the model, and a basic editor. TRAM uses Document Object Model (DOM) convention for parsing XML and XMI documents into a common data structure for further processing.

Figure 6.5: Fragment of SOM Ecore Metamodel
CHAPTER 6. TOOL SUPPORT FOR THE SOM APPROACH

6.4 SOM Composer

6.4.1 Component Design

The SOMi represent individual requirement instances captured from the input NLR. However, for analysis purposes, an overall perspective of the captured information is required. The SOM Composer matches and merges SOMi into a SON, which is a conceptual model that provides a comprehensive overview of the initial NLR, as well as an analysis mechanism that identifies requirements consistency (Section 5.6.5).

SOMs and SONs contain the same information, but their perspectives are complementary. While SOMs are defined as being requirement-centred, which allows the representation of each individual requirement and its components, SONs are products of SOMi composition and are concept-centred. This allows the user to understand exactly how concepts within an NLR relate and interact with each other. For example, SONs allow users to view all the actions that a particular agent performs within the system.

After filling in empty SOMi nodes during the Q&A process, all instances are lemmatised and transformed into lexemes. Lemmatisation is achieved by determining the uninflected form of the word, through the use of a dictionary and performing morphological analysis. In parallel the correspondence between grammatical number and instance cardinality is recorded (e.g. plural denotes many instances, while singular denotes one instance). Finally, complete SOMi are composed into a SON according to the SON composition rules (Section 5.6.4).

SON is a knowledge representation network created from directed graphs to analyse large numbers of requirements. SONs represent the textual requirements specification document in its entirety through the representation of the concepts and relationships it contains.

6.4.2 Implementation

The SOMi Composer takes as input a list of all SOMi and structures them in a SON, which is implemented as a hash map data structure. The SON hash map key is represented by a tuple containing the properties of a SOMi concept such as the lemma and concept type (e.g. Thing or Action). The SON hash map value contains the rest of the properties (e.g. name, id, POS, cardinality, etc.).

The structure for the SON XMI is based on the SON ECORE meta-model. Figure
6.5  NLR GENERATOR

6.5.1 Component Design

TRAM’s NL Generator is involved in the validation process. Its tasks are to check the consistency of the SON and offer domain experts with automatically generated natural language descriptions of the knowledge captured through the SOM approach. In this manner users are able to validate, or edit the captured knowledge.

Due to the inconsistencies of NLR documents and current limitations of NLP tools, validation of the SOM approach is crucial. First, the SON is checked for consistency. Because NLRs describe information systems based on processes, the resulted SON should be a traversable graph. However, if for some reason (e.g. use of synonyms, use...
of abbreviations, missing requirements) the SON becomes fragmented, this indicates a lack of model consistency. Consequently, the cause of the inconsistency needs to be further analysed and possibly discussed with domain experts.

Second, the SON is checked to validate its correspondence to the input NLR. Paraphrasing has been regarded as a viable solution for validating conceptual schemata derived from NLRs [RP92, MAA08]. This solution is motivated by the fact that natural language is the most appropriate communication language among users [CKO92]. In order to validate the SOMi, we propose a template filling approach using the NL generation rules. SOM patterns fit comfortably within templates as they were designed to capture common deep structures expressed through recurring surface structures. By surface structures we mean the syntactical form of the sentence, as described by grammatical rules. Deep structure relates to the meaning that a sentence conveys. For example, “John printed a report” and “The report was printed by John” have different surface structures but the same deep structure. They are also represented by the same Creation SOM.

SOMi are always paraphrased in active voice in order to be easy to read and understand the information currently captured. NL generation is performed according to the eight rules defined in Section 5.5. Rule 26 is used to generate natural language phrases from Change, Cognition, Creation, Motion, Perception, Contact, and Stative SOMs. Rules 27 and 28 are used to generate text from the Communication SOM, while Rule 29 is used to generate natural language phrases from Possession SOMs. Additionally, there are four rules for generating NLR containing behaviour, corresponding to the four implemented behaviour patterns: sequence (Rule 30), parallel split (Rule 31), synchronization (Rule 32), and XOR (Rule 33).

### 6.5.2 Implementation

The SON MDFS algorithm for consistency check is implemented as a recursive function that receives as parameters the number of concepts in the SON, a two-dimensional matrix that represents the SON graph, the position of a node, and a boolean array which remembers the taken paths.

For the correspondence validation, the NLR Generator paraphrases the SOMi using the NL Generation Rules specified in Section 5.5. These sentences are generated at run-time and are displayed in the user interface. Through the interface the user can modify the paraphrased requirements. The modifications trigger SOMi updates.
6.6 Summary

This chapter follows on from Chapters 4 and 5, which described the SOM Patterns and the SOM Translation Rules and their Application. These two chapters represent the SOM approach. The current chapter described TRAM, which is a software prototype designed and implemented to support the automatic construction of conceptual models through SOM approach.

TRAM’s software architecture is the result of the application of OO paradigm. It consists of four main components: NLR Parser, SOM Constructor, SOM Composer, and NLR Generator. The first three software components translate an input NLR into SON conceptual models by implementing the translation rules described in Chapter 5. The fourth component, is designed to check the consistency of the SON model, as well as the correspondence between the input NLR and the generated SON.
Chapter 7

Validation and Practical Evaluation

7.1 Introduction

This chapter is divided in two parts. The first part presents the empirical validation of the SOM approach. This addresses the research question defined in Chapter 1. The second part, describes the practical evaluation of the SOM approach implementation, i.e. TRAM.

7.2 SOM Validation

This section describes the experiment to validate the semantic-driven approach. It starts by laying out the objectives and setting of the experiment. The remainder of this section covers the approach, results of the experiment, and analysis. Finally, a summary of the experiment is included at the end of the section.

7.2.1 Objectives

The experiment aimed to empirically validate the proposed semantic-driven approach of constructing comprehensive conceptual models from NLRs. In order to perform this experiment, undergraduate and postgraduate computer science students have participated as research subjects. The goal definition for the experiment is listed below:

Object of study: Transformation approach of NLRs into conceptual models using SOMs

Purpose: To undertake an experiment to validate the proposed SOM approach
Focus: To evaluate the effectiveness of the SOM approach in terms of producing comprehensive conceptual output models compared to the manual approach.

Perspective: From the point of view of the users of the analysis approach.

Context: In the context of users who are not domain experts.

The focus of this study is the main research question derived from the identified research problem defined in Chapter 1. Furthermore, in order to identify and focus on the issues to be investigated, two propositions were developed. The collected experiment data is compared against the propositions in order to support or reject them, therefore, supporting or rejecting the hypothesis.

The following research question and propositions have been addressed in this experiment:

RQ: Can the SOM approach produce comprehensive conceptual models during the analysis process?

P-1: The semantic-driven analysis process will guide the users to identify elements missing from the NLR.

P-2: The semantic-driven analysis process will guide the users to identify and represent elements that might be overlooked during the manual process.

7.2.2 Experiment

Approach

The experiment was conducted in the School of Computer Science at the University of Manchester with the participation of both undergraduate and postgraduate students. In total, six computer science students (four undergraduates and two postgraduates) volunteered to take part in the experiment. On average, a participant spent 52 minutes performing the experiment.

Participants were given a text containing NLR statements describing a library system [Cal94] and asked to perform two tasks. The first task was to extract functional requirements from the input NLR using their current knowledge of requirements analysis. Before starting the second task, the participants were given a brief overview of the SOM models and some materials in order to perform the SOM analysis. The materials consisted of a table containing verbs from the input NLR text and what SOM each
Participant | Set, | Task1 | Task2 | S_{c_{Task2}} | S_{c_{Task1}} | (%) | S_{c_{Task2}} | S_{c_{Task1}} | (%)
--- | --- | --- | --- | --- | --- | --- | --- | --- | ---
P1 | S_t | 16 | 24 | 8 | 7 | +50% | +77.78%
P2 | S_t | 16 | 24 | 8 | 7 | +50% | +100%
P3 | S_t | 23 | 30 | 7 | 9 | +30.43% | +128.57%
P4 | S_t | 22 | 27 | 5 | 2 | +22.73% | +14.29%
P5 | S_t | 21 | 23 | 2 | 2 | +9.52% | +15.38%
P6 | S_t | 21 | 26 | 5 | 1 | +23.81% | +6.25%
Average | S_t | 21 | 26 | 5 | 1 | +31.08% | +57.05%

Table 7.1: Quantitative results of the SOM validation experiment show the improvement in requirements completeness from Task 1 to Task 2.

The second task was to extract functional requirements from the input NLR, by identifying every verb and attempting to fully instantiate the necessary SOM patterns. Participants were instructed to ask questions whenever they felt the need to clarify the requirements or need extra information. In order to provide additional information in case of further elicitation questions, the researcher acted as domain expert.

**Results**

The interpretation criteria for measuring the validity of propositions P1 and P2 is based on the completeness of requirements extracted using the SOM approach (Task 2) compared against the completeness of the previously extracted requirements (Task 1) by each participant. By comparing the number of concepts captured by requirements from the two tasks against each other, one can observe the measurable size differences between the two sets. The set with a higher number of concepts will be deemed more complete, therefore comprehensive.

In order to compare the two sets of requirements, first the sets of identified concepts (i.e. THINGS and ACTIONS) in each task are established. Then, for each participant, the set of new concepts from Task 2 (S_{c_{Task2}}) that are not present in Task 1’s
7.2. SOM VALIDATION

<table>
<thead>
<tr>
<th>Participant</th>
<th>Set&lt;sub&gt;c&lt;/sub&gt;</th>
<th>Task1</th>
<th>Task2</th>
<th>S&lt;sub&gt;cTask1&lt;/sub&gt;</th>
<th>S&lt;sub&gt;cTask2&lt;/sub&gt;</th>
<th>S&lt;sub&gt;cNew&lt;/sub&gt;</th>
<th>S&lt;sub&gt;cMissing&lt;/sub&gt;</th>
<th>S&lt;sub&gt;cOverlooked&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>S&lt;sub&gt;t&lt;/sub&gt;</td>
<td>16</td>
<td>24</td>
<td>8</td>
<td>8</td>
<td>11</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>S&lt;sub&gt;a&lt;/sub&gt;</td>
<td>9</td>
<td>16</td>
<td>7</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td>S&lt;sub&gt;t&lt;/sub&gt;</td>
<td>16</td>
<td>24</td>
<td>8</td>
<td>8</td>
<td>1</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>S&lt;sub&gt;a&lt;/sub&gt;</td>
<td>7</td>
<td>14</td>
<td>7</td>
<td></td>
<td>11</td>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>P3</td>
<td>S&lt;sub&gt;t&lt;/sub&gt;</td>
<td>23</td>
<td>30</td>
<td>7</td>
<td>10</td>
<td>3</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>S&lt;sub&gt;a&lt;/sub&gt;</td>
<td>7</td>
<td>16</td>
<td>9</td>
<td></td>
<td>11</td>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>P4</td>
<td>S&lt;sub&gt;t&lt;/sub&gt;</td>
<td>22</td>
<td>27</td>
<td>5</td>
<td>7</td>
<td>3</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>S&lt;sub&gt;a&lt;/sub&gt;</td>
<td>14</td>
<td>16</td>
<td>2</td>
<td></td>
<td>4</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>P5</td>
<td>S&lt;sub&gt;t&lt;/sub&gt;</td>
<td>21</td>
<td>23</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>S&lt;sub&gt;a&lt;/sub&gt;</td>
<td>13</td>
<td>15</td>
<td>2</td>
<td></td>
<td>7</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>P6</td>
<td>S&lt;sub&gt;t&lt;/sub&gt;</td>
<td>21</td>
<td>26</td>
<td>5</td>
<td>8</td>
<td>4</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>S&lt;sub&gt;a&lt;/sub&gt;</td>
<td>16</td>
<td>17</td>
<td>1</td>
<td></td>
<td>9</td>
<td>0</td>
<td>9</td>
</tr>
</tbody>
</table>

Table 7.2: Qualitative results of the SOM validation experiment supports propositions P-1 and P-2

set (S<sub>cTask1</sub>) is recorded as a new set S<sub>cNew</sub>. Within S<sub>cNew</sub>, if the concept is not a word from the text (i.e. it is the result of elicitation questions), then it is part of the missing concept set (S<sub>cMissing</sub>). Otherwise, the concept is part of the overlooked concept set (S<sub>cOverlooked</sub>). Equation 7.1 represents the previously described relations:

\[
S_{cTask2} \setminus S_{cTask1} = S_{cNew} = S_{cMissing} \cup S_{cOverlooked}
\] (7.1)

Table 7.1 summarises the quantitative analysis of the experiment. For each participant, there is a count of the number of Things and Actions in each set in both Task 1 and Task 2, followed by the set size difference. The final column, represents the percentage by which the sets in Task 2 are larger than sets in Task 1. Strictly from a quantitative point of view, the results are highly encouraging. All participants have benefited from using the SOM approach, as both their sets have gone up in size on average, with roughly 31% for Things and 57% for Actions.

By analysing the data from a qualitative point of view, the results offer additional positive insight. Table 7.2 summarises the qualitative analysis of the experiment. The third and fourth columns represent the size of concept sets identified during the two tasks. The fifth column, represents the size difference between sets from different tasks (i.e. what is the size difference of the set of concepts from Task 1 to Task 2). The sixth column, S<sub>cNew</sub>, represents the actual set difference between S<sub>cTask2</sub> and S<sub>cTask1</sub>.

\[\text{Where } c \text{ could be replaced by either } t \text{ (Things) or } a \text{ (Actions)}\]
(i.e. what concepts are present in Task 2 that are not present in Task 1).

Participants have introduced external concepts to the analysis through their assumptions, during Task 1. Whereas, during Task 2, participants have been more likely to identify and use concepts from the input NLR and ask elicitation questions pertinent to the domain. Therefore, it can be observed that the size difference does not always correlate with the concept difference (i.e. columns 5 and 6 are not identical). Even so, the concepts identified through SOM analysis (column 6) is always larger in size than the concepts found through manual analysis, suggesting that the SOMs are indeed helpful in guiding the analysis process.

Finally, the last two columns reveal what is the split between missing concepts (i.e. concepts not present in the NLR, but identified through further elicitation) and overlooked concepts (i.e. present in the NLR but ignored by the participants). These results strongly suggest that the SOM approach, supports not only the identification of missing concepts, but also the identification of concepts that would otherwise be overlooked by analysts, therefore supporting propositions P-1 and P-2. It can be observed that all new verb concepts are overlooked, as this is a consequence of using the SOM approach, which focuses on expanding all actions present in the NLR.

In conclusion, the following observations can be made from the analysis of the experimental data:

1. In both Tasks 1 and 2, participants have made certain assumptions, resulting in the addition of external concepts that were not reflected in the input NLR. The SOM approach has diminished to a certain extent these assumptions, but did not eradicate them altogether. This is in fact a beneficial feature, as the SOM approach is not a mechanism meant to stifle the creativity and expertise of its users, but rather to guide them in systematically identifying concepts.

2. The semantic-driven approach has enhanced the participants’ observation of the problem domain by guiding them to ask further elicitation questions in order to fully instantiate the SOMs. Using the equation below, it was calculated that on average, 35.48% of the newly identified concepts ($S_{t\text{New}}$) were missing from the input NLR. This finding supports proposition P-1.

$$S_{t\text{Missing}}(\%) = \frac{\sum_{i=1}^{n} S_{t\text{Missing}}}{S_{t\text{New}}} \cdot \frac{1}{n} \quad (7.2)$$

Where $n$ equals the number of participants in the experiment.
3. By making SOM structures explicit, participants were guided to identify concepts present in the NLR that were overlooked by them during Task 1. Using Equation 7.3, it was calculated that on average, 64.52% of the newly identified Thing concepts ($S_{t_{\text{new}}}$) were overlooked by participants during Task 1. In terms of Action concepts ($S_{a_{\text{new}}}$), 56.78% of actions were overlooked (using Equation 7.4). This finding supports proposition P-2.

$$S_{t_{\text{Overlooked}}} = \frac{\sum_{i=1}^{n} S_{t_{\text{Overlooked}}}}{S_{t_{\text{New}}}}$$ \hspace{1cm} (7.3)

$$S_{a_{\text{Overlooked}}} = \frac{\sum_{i=1}^{n} S_{a_{\text{Overlooked}}}}{S_{a_{\text{Task2}}}}$$ \hspace{1cm} (7.4)

In conclusion, the experimental data supports both propositions P-1 and P-2, that the SOM approach guides the users to identify elements missing from the NLRs and also identify and represent elements that might be overlooked during unguided analysis. Therefore, the original hypothesis presented in Chapter 1, that a semantic-driven approach based on verb classification and modelling constructs can improve the analysis process with regards to model completeness is supported.

In the following sections, the TRAM validation method is described, followed by its evaluation against related works such as pattern-based semantic and ontology-based driven approaches, as well as linguistic approaches.

### 7.3 Practical Evaluation of TRAM

Transformation approaches are evaluated based on the types and qualities of their outputs. However, the evaluation of requirements models is a challenging task. There is no established standard evaluation method in existence. The main reason behind this challenge is the fact that one of the purposes of requirements models is to serve as communication devices between technical and non-technical stakeholders [CKO92]. Consequently, models obtained in such transformations are not good or bad, but rather useful or less useful [Fow97], which makes them difficult to assess.

The thesis of this project is that using linguistic patterns based on both syntactic and semantic information will lead to the creation of more accurate conceptual models. In order to assess accuracy, however, a better understanding of how different types of
knowledge contribute to the validation process needs to be developed. The analyst understands the syntax and semantics of the model. However, only the domain expert possesses the specific domain knowledge that would enable the validation of the output model. In consequence, an appropriate method of evaluating the transformation should take into account both types of knowledge.

Rolland and Proix [RP92] assert that software modelling entails, both domain knowledge and modelling knowledge. Therefore, an approach needed to be developed that could take both types of knowledge into account. First, there is a distinction made between the two types of knowledge. Each has its own experts i.e. the domain expert (DE) and the modelling expert (ME). In most of the cases they are different individuals, with different expertise and goals. For example, DEs are interested that the software models precisely describe the NLRs. MEs are interested that software models comprehensively cover NLR concepts. Furthermore, each individual has a different world view, that may skew results in one way or another.

The modelling expert’s knowledge is useful in identifying modelling concepts from the NLR necessary for creating the analysis models. For example, if the target analysis models are UML class diagrams, the analyst will be looking for concepts expressed through nouns that can be represented as classes, actions expressed through verbs that can be represented as methods, and properties expressed through adjectives that can be represented as attributes.

The domain expert’s knowledge is useful in validating that the model representation is actually reflecting the domain’s requirements that would fulfil the user’s needs. However, there are cases in which the domain expert does not have the necessary knowledge to understand the model representation. In these cases then, the analyst needs to translate the model in natural language so that the domain expert may validate it.

A standard measure of relevance has been adapted from information retrieval [GS96], based on the precision and recall metrics, in order to assess how accurate TRAM is [HG00]. Information retrieval metrics such as precision and recall are widely accepted as standard performance measures. Precision can be used as a measure of correctness, whereas recall can be used as a measure of completeness. In order to achieve a comprehensive evaluation both types of knowledge need to be addressed. Consequently, two research questions have been formulated for this evaluation:

**RQ 1:** From a DE perspective, how correct is the conceptual model in representing the original NLRs (related to precision)? (Section 7.3.1)
RQ 2: *From an ME perspective, how complete is the conceptual model in covering the original NLRs (related to recall)?* (Section 7.3.2)

The two metrics are then composed into an accuracy metric which is a percentage that describes the usefulness of a model. The interpretation of 100% accuracy is the following: all the requirements captured by the model are correct; moreover, all the concepts that could be represented in the conceptual model have been extracted from the original NLR. Therefore, a third research questions has been formulated:

RQ 3: *Combining both DE and ME perspectives, how accurate is the conceptual model in capturing and describing the original NLRs?* (Section 7.3.3)

Validation of the SOM approach occurred in two phases. In the first phase, the SOM approach was evaluated by both domain and modelling experts. Domain experts assessed the correctness of the SON model, whereas modelling experts assessed the completeness of the output SON model. Using the two metrics the accuracy of the SON model was calculated. By providing a way of calculating the accuracy the SON model can be evaluated against related approaches that output different types of requirements models (e.g. UML class diagrams). Thus, in the second phase, the SOM approach was compared with eight tools over five input NLRs (Appendix A.1). Five modelling experts (MEs) and two researchers acting as domain experts (DEs) have been involved in the evaluation process. The MEs have assessed the NLR examples independently, thus providing five sets of completeness measures. The two DEs have assessed the seven related approaches independently, after which they aggregated their results so that there was only one correctness measure for each output model. In essence this procedure simulates the fact that there might be multiple viable modelling solutions (many completeness measures), however there is only one correct interpretation of the requirements (one correctness measure).

### 7.3.1 Correctness: Domain Expert Evaluation

**Approach**

Figure 7.1 depicts the domain expert evaluation process, which measures the correctness of a SON model. This method consists of the following steps. TRAM automatically generates a paraphrased NLR from the SON model. The domain expert then reviews the paraphrased NLR and gives a feedback to indicate if the paraphrased
CHAPTER 7. VALIDATION AND PRACTICAL EVALUATION

Figure 7.1: Domain expert evaluation process of SON Correctness

NLR correctly matches the original NLR. The domain expert will state whether a requirement is a correct match (when cardinality is also correct), a partial match (when cardinality is incorrect), a mismatch (when the paraphrased requirement is incorrect), or unclear. For each requirement paraphrase a complete match will be marked with two points, a partial match with one point, a mismatch with zero, and unclear is not marked, only recorded.

In order to achieve a comprehensive evaluation of the proposed approach, requirements are evaluated from two perspectives. First, the description of the system behaviour, and second, the cardinalities of each element described in the requirement. Consequently, each requirement has a weight of two.

The correctness of the captured requirements is measured by generating NLR from the captured information and feeding it back to the domain expert for evaluation [BDADSS11]. For example, TRAM would ask if the requirement “One ATM updates one balance with one amount” is fully correct, partially correct, incorrect, or unclear. The formula depicted in Equation 7.5 is used to calculate the correctness of the output model, i.e. the percentage of captured requirements that are correct. This correctness is measured by multiplying by two the number of requirements correctly matched adding the result with the number of partially matched requirements over the number of all requirements covered multiplied by two:

\[
Correctness = \frac{2 \times N_R + N_{PR}}{2 \times N_T}
\]  

(7.5)

where \(N_R\) and \(N_{PR}\) respectively denote the number of correct matches and partial matches; \(N_T\) represents the sum of correct matches, partial matches, mismatches, or
7.3. PRACTICAL EVALUATION OF TRAM

Two researchers acted as domain experts during the correctness evaluation process. The first was a senior researcher from University of Oviedo, Spain, specialised in Model Driven Development. The second was a third year PhD candidate from the University of Manchester, United Kingdom, specialising in requirements transformation and traceability within Model Driven Development.

Each expert was given four TRAM generated NLRs to review against the original four input NLRs. The experts then recorded the number of correct matches, partial matches, mismatches, or unclear requirements for each paraphrased NLR against the original NLR document. After both experts finished the initial round of individual assessment, a second round began, in which both experts compared their results in order to achieve agreement. This round was carried out in order to reduce the bias and subjectivity of the individual interpretations. Around 85% of instances were agreed upon from the start (including correct and partial matches, as well as incorrect or unclear requirements). The rest, approximately 15% of instances, needed negotiations between experts to resolve the conflicting views.

The final results were recorded and the correctness was then calculated for each input NLR, using Equation 7.5. The results are summarised in Table 7.4. The correctness measure indicates how well the output model reflects the requirements described in the input NLR, in terms of relationships and cardinalities.

7.3.2 Completeness: Modelling Expert Evaluation

Approach

Figure 7.2 depicts the modelling expert evaluation process, which measures the completeness of a SON model. This measurement is achieved by asking a modelling expert to review the original NLR and select the relevant elements from the text that would be useful in the creation of a conceptual model (e.g. UML class diagram). Once the human modeller finishes this task, the results are compared against the output SON. The result will indicate whether the SON model contains all the concepts indicated by the human modeller and if not, which concepts the TRAM approach has failed to capture.
In this way the completeness evaluation process takes into account the variability of human decision making. For example, two human modellers will obtain different results even though the input NLR was identical. This is due to bias, experience, and breadth of domain and modelling knowledge. As a consequence, the completeness percentage will indicate how well the SOM approach compares to human modellers.

The completeness metric (Equation 7.6) is used to calculate the completeness of concepts captured in the SON model. This completeness metric is measured by dividing the number of concepts identified in the SON model to the total number of concepts that the modelling expert has identified:

\[
\text{Completeness} = \frac{N_C}{N_C + N_{MC}} \quad (7.6)
\]

where \(N_{MC}\) refers to the number of concepts identified by the modelling expert but not represented in the SON model.

Equation 7.6 is used to calculate completeness for both Action concepts (\(\text{Recall}_A\)) and Thing concepts (\(\text{Recall}_T\)), to allow us to understand the concept coverage for both actions and things. Both actions and things have equal weights, for that reason the final total completeness is calculated as an arithmetic mean (Equation 7.7).

\[
\text{Completeness}_{\text{final}} = \frac{\text{Completeness}_A + \text{Completeness}_T}{2} \quad (7.7)
\]

where \(\text{Completeness}_A\) refers to the completeness of the actions of TRAM and \(\text{Completeness}_T\) refers to the completeness of the things.
Results

The NLR examples were classified into two groups based on practical considerations. The average size of the input NLRs was around 220 words. For such a document, an expert took on average 20 minutes to analyze. This interval represented the time needed by an expert to read and comprehend the document and select the relevant concepts from the text; no modeling was involved in this interval. Furthermore, some studies contained better quality output models, or multiple outputs. Lastly, variation of the problem domain was taken into account. Consequently, the initial set of nine NLRs was split into two priority groups. The first priority group consisted of four example NLRs from various domains such as retail, library, and business, and yielded seven output models that could be compared against. The, second priority group consisted of the remaining five example NLRs, from domains such as aviation, business, and banking which yielded five output models.

Six modelling experts were invited to participate in the completeness evaluation. Experts were asked to extract elements to build any type of conceptual analysis model, without restriction to the way they perform analysis. All the experts reported choosing OOA, thus identifying classes, attributes, methods etc. Out of the five modelling experts, one analysed all nine examples and identified the necessary concepts to build conceptual analysis models. Due to time constraints, four experts analysed only the first priority group containing four NLR documents. One of the experts has not been able to carry out the task altogether. The modellers’ experience ranged from an average of five years’ experience for Experts 3, 4, and 5 to over 20 years of experience for Experts 1 and 2 in conceptual modelling. In this chapter the results for the first priority group are discussed.

The output SON model for each input NLR was compared against each of the modelling experts’ analysis outcome. The result of the comparison recorded the number of concepts covered in the SON model and the number of missing concepts. Based on the recorded data, the completeness for each SON model was calculated against each expert. Finally, completeness is calculated for each SON component using Equation 7.6 (i.e. actions and things). Table 7.5 summarises the results. The completeness measure indicates how many of the concepts deemed important by the human modellers are covered by the output model describing the input NLR, in terms of actions and things.
7.3.3 Accuracy: Combining Correctness and Completeness

Approach

Conceptual models are human artefacts that are not right or wrong, but rather more or less useful [Fow97]. Consequently, the evaluation method presented in this thesis calculates the accuracy of output models as percentages out of human accuracy. That is to say that the models created by the modelling experts are considered the gold standard and the purpose of the evaluation is to verify how close the output models are to human modellers. As a result, for each modelling expert the accuracy measure of each output changes to reflect how close the output model is to the proposed solution.

Results

Both correctness and completeness are considered to have equal weight for the purpose of this study. Consequently, accuracy is calculated as the harmonic mean between correctness and completeness (Equation 7.8). Because there are different completeness measures depending on the modelling experts’ choices, there are also different accuracy percentages (Table 7.6).

\[
\text{Accuracy} = \frac{2 \times \text{Correctness} \times \text{Completeness}_{\text{final}}}{\text{Correctness} + \text{Completeness}_{\text{final}}}
\]  

(7.8)

7.4 Comparison With Related Tools

Eight tools over five examples are described in this section. Each of the eight output models were analysed using the present evaluation method (Figure 7.3). Table 7.3

Figure 7.3: Applying the same evaluation process to related approaches
7.4. COMPARISON WITH RELATED TOOLS

presents the links between approaches and examples. Examples 1 and 2 describe library systems and contain two respectively three approaches. Examples 3, 4, and 5 contain one approach each, from retail, business, and aviation domains respectively. Furthermore, Harmain and Gaizauskas’ CM-Builder approach [HG00], presented in Example 1, is an ontology-based semantic-driven approach, whereas CIRCE [AG06] is an pattern-based semantic-driven approach. The rest of the approaches are linguistic. In the following sections, the evaluation results are presented. The full NLRs are presented in Appendix A.

<table>
<thead>
<tr>
<th></th>
<th>Library</th>
<th>Retail</th>
<th>Business</th>
<th>Aviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example 1</td>
<td>Example 2</td>
<td>Example 3</td>
<td>Example 4</td>
<td>Example 5</td>
</tr>
<tr>
<td>Linguistic</td>
<td></td>
<td></td>
<td></td>
<td>CIRCE [AG06]</td>
</tr>
<tr>
<td>Pattern</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ontology</td>
<td>CM-Builder [HG00]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7.3: Approaches and examples evaluated

7.4.1 Correctness

The goal of this method is to measure the correctness of the output models in their representation of the input NLRs. This method consists of the following steps. The model validator automatically generates paraphrased questions from the output model. The domain experts then review the paraphrased questions and give an answer to indicate if the paraphrased question matches the original NLR. The domain expert will state whether a question is a correct match (cardinality is also correct), a partial match (cardinality is incorrect), a mismatch (question is wrong), or unclear. For each paraphrased question, a correct match will be assigned two points, a partial match with one point, a mismatch with zero, and unclear questions are not marked, only recorded. Equation 7.5 is used to compute the correctness measure.

The correctness indicates how well the output model reflects the requirements described in the input NLR, in terms of relationships and cardinalities. Out of the eight outputs that were evaluated, two were created manually, respectively Callan [Cal94].
CHAPTER 7. VALIDATION AND PRACTICAL EVALUATION

Table 7.4: Correctness measures for related approaches and TRAM

<table>
<thead>
<tr>
<th>Example 1</th>
<th>Example 2</th>
<th>Example 3</th>
<th>Example 4</th>
<th>Example 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>CM-Builder [HG00]</td>
<td>7.69</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Callan [Cal94]</td>
<td>75</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RACE1 [IA10]</td>
<td></td>
<td>38.89</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RACE2</td>
<td></td>
<td>42.86</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RACE3 [SBM12]</td>
<td></td>
<td>44.44</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVO [LDP04]</td>
<td></td>
<td></td>
<td>45.45</td>
<td></td>
</tr>
<tr>
<td>Harmon [HW97]</td>
<td></td>
<td></td>
<td></td>
<td>75</td>
</tr>
<tr>
<td>CIRCE [AG06]</td>
<td></td>
<td></td>
<td></td>
<td>80</td>
</tr>
<tr>
<td>TRAM</td>
<td>60</td>
<td>94.44</td>
<td>72.41</td>
<td>54.69</td>
</tr>
</tbody>
</table>

and Harmon [HW97]. Understandably, both these models have high correctness measures, specifically 75%. However, not even these approaches are 100% precise, either because the NLR was misinterpreted or the modelling expert failed to put the required information in the output model. CIRCE achieves 80% correctness due to its input (CIRCE requires as input, in addition to the NLR, a glossary that describes, classifies, and provides definitions of all the domain and system specific terms used in the requirements).

Overall, TRAM performed better than the compared tools, being outperformed only by the two manual approaches and CIRCE (Table 7.4). The manual approaches are dependent on the expertise of the modeller. Human modellers are good at interpreting and inferring properties of missing instances. For example, an automated tool would not know how to link a pronoun to a concept in the text and assign it the correct cardinality, unless it would have a way of correctly solving co-referencing issues. In CIRCE’s case, the modeller’s expertise is also used in the creation of supplementary models that define terms from the NLR. Therefore, CIRCE’s input requires a considerable amount of effort to create. TRAM has shown potential to close the gap between these approaches through its SOM approach which contains an embedded Q&A process. Consequently, TRAM achieved 94.44% precision in one instance (Example 2).

7.4.2 Completeness

The goal of this method is to measure the completeness of the concepts represented by the related approaches’ output models. At this point every one of the input NLRs has been analysed by the five modelling experts. Each set of expert results is compared against the related approaches’ outputs. Equations 7.6 and 7.7 are used to compute the individual completeness measures for actions and things and the final completeness
### Table 7.5: Completeness measures for related approaches and TRAM (A - Action Completeness, T - Thing Completeness, R - Average Completeness for both Things and Actions)

The results indicate how many concepts identified by the human modeller have been covered by the output models. Consequently, the completeness percentage indicates how close the output models are to the models obtained by the human modellers.
In contrast to correctness, TRAM’s average completeness is better than Callan’s and Harmon’s (Table 7.5). An explanation for this fact is that automated tools tend to accumulate comprehensive sets of elements without discerning among them. For example, RACE3 achieves the highest average completeness percentage by identifying all the nouns and verbs within the input NLR.

Overall, TRAM performed better than most of the approaches both manual and automated, being outperformed in two instances by RACE3 and SVO (Table 7.5). This is due to the fact that TRAM performs concept identification by relying on a linguistic approach rather than a purely lexical one (e.g. identifying all nouns in the input NLR). For example, if a sentence is not well formed and the parser creates an erroneous parse tree, then TRAM might not identify some relevant concepts.

### 7.4.3 Accuracy

The results presented in Table 7.6 indicate how close the evaluated approaches have come to replicating the requirements models created by the modelling experts. The higher the accuracy percentage the better. Accuracy analysis can only be performed within the groups that share the input examples (i.e. (CM-Builder, Callan, TRAM), (SVO, TRAM), etc.). Analysing the Average column of Table 7.6 two conclusions can be drawn:

- TRAM performs better than all the automated approaches and is on par with the manual approaches (Callan, Harmon);
- The accuracy scores obtained by the manual approaches support the fact that for each NLR input every modelling expert will come up with a different solution.

Overall, the results presented in Table 7.6 suggest that TRAM is a strong candidate for performing conceptual modelling by providing (1) better results than the related automated approaches and (2) comparable results to the manual approaches in a shorter time interval.

### 7.5 Comparison With Related Approaches

In this section, the SOM approach is compared with related approaches in the literature that have not provided examples for their evaluation. Instead, the comparison is made based on the three evaluation criteria defined in Chapter 2: requirements preprocessing, transformation approach, and validation.
7.5. COMPARISON WITH RELATED APPROACHES

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Callan</td>
<td>82.98</td>
<td>66.32</td>
<td>51.56</td>
<td>53.48</td>
<td>52.17</td>
<td>61.30</td>
</tr>
<tr>
<td>TRAM</td>
<td>68.04</td>
<td>69.97</td>
<td>40</td>
<td>52.14</td>
<td>52.14</td>
<td>56.46</td>
</tr>
<tr>
<td>RACE1</td>
<td>43.75</td>
<td>41.72</td>
<td>41.48</td>
<td>41.48</td>
<td>35.9</td>
<td>40.87</td>
</tr>
<tr>
<td>RACE2</td>
<td>46.16</td>
<td>43.9</td>
<td>43.64</td>
<td>43.64</td>
<td>37.5</td>
<td>42.97</td>
</tr>
<tr>
<td>RACE3</td>
<td>57.45</td>
<td>54.37</td>
<td>60.44</td>
<td>54.2</td>
<td>57.97</td>
<td>56.89</td>
</tr>
<tr>
<td>TRAM</td>
<td>75.22</td>
<td>77</td>
<td>80.04</td>
<td>80.04</td>
<td>65.38</td>
<td>75.53</td>
</tr>
<tr>
<td>SVO</td>
<td>58.82</td>
<td>54.33</td>
<td>54.15</td>
<td>55.37</td>
<td>56.12</td>
<td>55.76</td>
</tr>
<tr>
<td>TRAM</td>
<td>71.61</td>
<td>71.61</td>
<td>69.58</td>
<td>71.61</td>
<td>56.76</td>
<td>68.23</td>
</tr>
<tr>
<td>Harmon</td>
<td>81.82</td>
<td>65.57</td>
<td>63.83</td>
<td>50</td>
<td>53.48</td>
<td>62.94</td>
</tr>
<tr>
<td>TRAM</td>
<td>69.42</td>
<td>68.69</td>
<td>65.69</td>
<td>67.31</td>
<td>65.35</td>
<td>67.29</td>
</tr>
<tr>
<td>CIRCE</td>
<td>79.28</td>
<td>55.31</td>
<td>62.32</td>
<td>58.18</td>
<td>65.57</td>
<td>64.13</td>
</tr>
<tr>
<td>TRAM</td>
<td>62.3</td>
<td>61.34</td>
<td>59.75</td>
<td>59.88</td>
<td>62.03</td>
<td>61.06</td>
</tr>
</tbody>
</table>

Table 7.6: Accuracy measures for related approaches and TRAM

7.5.1 Requirements Processing

The SOM approach takes as input NLR documents. There are no restrictions on the format. Lexical, syntactic, and semantic analysis are performed, including tokenization, sentence splitting, POS tagging, parse tree generation through probabilistic context free grammar, and typed dependency generation. The analysis is carried out for each individual sentence.

From the input’s point of view the SOM approach does not diverge from other related approaches in the literature. From a requirements processing point of view, the approach performs a comprehensive analysis, making use of advanced NLP techniques [HG00], [Mic96], [IO06], [FKM+07], [SP10].

7.5.2 Transformation Processes

The SOM approach employs an indirect transformation type, similar to others [HG00], [Mic96], [IO06], [YBL10], [FKM+07]. Indirect transformation refers to having at least one intermediate model between the NLRs and requirements models, while direct transformation has none. Early direct transformation attempts saw the correlation of natural language fragments to elements of conceptual models [Abb83, Che83] or to code directly [SHE89]. However, this implies that the NLRs used as input are created by clients who know what they want and they can express their thoughts in written
descriptions that contain complete, consistent requirements, comprehensively describing the desired system. In reality, clients do not know what they want until later in the software development stage and NLRs are incomplete [BJ95]. Moreover, errors undetected at requirements stage are the most difficult and costly to repair [Dav93]. Direct transformations would be ideal since they provide direct traceability and validation properties, easy maintainability and evolution. In order for direct transformation to work, it requires that texts follow certain syntactical guidelines such as: (1) sentences must be declarative, (2) each sentence must be simple (a single subject and a single main verb), (3) the verb must be transitive, (4) and the terminology restricted (no synonyms in order to ensure uniformity), (5) the description needs to be complete and correct.

Indirect transformation, on the other hand, contains an overhead for validation and verification, and there are higher chances of losing information [YBL11]. Still, the complexities of NLRs suggest that intermediate models play a key role in achieving successful automated transformation. Approaches that employ intermediate models may address issues such as requirements incompleteness, while at the same time being independent from a specific NLP technique or tool. Furthermore, indirect transformation offers the flexibility to map onto different conceptual models, essentially serving as a bridge between informal (NLR) and formal (requirements models) representations.

The SOM approach outputs a SON that represents the knowledge captured from the input NLR. The advantage of outputting SONs instead of directly outputting analysis models is re-usability. For example, the SOM approach does not need to adapt to whatever modelling language is used for outputting analysis models. Instead, the focus is only on creating a mapping between the SOM meta-model to the target language’s meta-model. This way the SOM approach can be used in concordance with any conceptual modelling language for software development such as Business Process Modelling notation (BPMN), Entity-Relationship Diagrams (ERD), Integration DEFinition (IDEF), Unified Modelling Language (UML) etc.

The reason why creating the mapping between the SOM meta-model and other conceptual modelling language is straightforward, lies in the nature of SOM’s meta-model. As previously described in Section 3.6, the SOM meta-model was constructed around the fundamental ontological constructs, from which all other constructs are derived [WW95]. As a result, all the elements within the SOM meta-model will have correspondents in the target meta-models.

There are a number of similarities between the SOM approach and other approaches
such as Diaz et al.’s action transformation patterns to describe semantic context [DPM05], Kamalrudin’s abstract interaction descriptions through essential use cases [KGH10], or Mich’s NL-OOPS, which builds a semantic network to represent domain knowledge [Mic96]. From a linguistic perspective the transformation process carried out by the SOM approach is based on rules and cases, similar to [FKM +07]. However, each approach has its limitations. For example there are issues when dealing with large system specifications, as the number of proposed candidates becomes high [HG00, OLR01]. Furthermore, the knowledge bases are manually built, meaning they are less comprehensive, not easily updatable, and difficult to scale up [SPKB09]. Moreover, the quality of the created conceptual graphs greatly depends on the quality of used text documents. In some cases the conditions for applying transformation rules are strict and it is not clear what would happen in the event that NL does not conform to those rules [DPM05]. Moreover, patterns are built manually without a clear scope which limits comprehensiveness of the approach [KGH10, KHG11]. On one hand, there are approaches that do not use semantic information in their transformations and the impact of gaps in the knowledge base is unknown being unable to detect omissions [DM07, Mic96]. On the other hand, there are approaches that rely heavily on complex glossaries, which are difficult to create and maintain [AG06, AG97, GKC07], while others rely heavily on NLP tools [FMP11], which do not cope well with NL issues such as ambiguity and omissions. Lack of evolution maintainability [IO06] and restricted input [KP11] are additional issues with current approaches.

The SOM approach avoids these limitations by relying on a small but comprehensive set of primitive conceptual models (SOMs) that provide additional semantic structure to the input NLRs and provide the ability to identify ambiguity and omissions in the textual inputs.

7.6 Threats to Validity

First, the small number of examples that the modelling experts analysed is a potential threat to the validity of the evaluation process. The decision to divide the sets of examples into to sets was taken from a practical point of view. The experts voluntarily offered to take on the analysis in their free time, so the decision to maximise this time was taken. On average, an hour and a half was necessary to go through the four examples. The expert that processed all nine input NLRs needed approximately three hours to complete the analysis.
Second, the analysis of models is a highly biased process. Although attempts to formalise the evaluation procedure through correctness and completeness measures were made to minimise the bias effects, the author acknowledges that there are still issues when evaluating the generated paraphrases. These issues rise from the individual interpretations of the text by the experts and manifests itself during the model validation.

7.7 Summary

This chapter has described the evaluation methods undertaken to validate the SOM approach. First, the validation method is established and the evaluation criteria are described in detail. Second, the effectiveness of the SOM approach is evidenced by comparison to related tools and approaches. Both evaluations support the proposed thesis of this research, which is that extraction of textual elements from the input NLR using linguistic patterns based on both syntactic and semantic information can support the software development process. Results indicate that the SOM approach leads to the creation of more accurate conceptual models than the current state of the art with minimal effort from the users. Finally, potential threats to the validity of the evaluation are summarised.
Chapter 8

Conclusions

In this thesis, an approach for automatic construction of requirements models from NLR documents to support early stages of software development has been defined. Central to this approach is a novel transformation approach, whose foundations are based on SOMs. The SOM approach has been evaluated empirically and theoretically. The empirical evaluation was conducted through examples of different domain input NLRs to test the utility and applicability of the approach. The theoretical evaluation was conducted through comparison against related work. This final chapter outlines the contributions and limitations of the research, as well as providing recommendation for future work.

8.1 Challenges

The following challenges have been identified with regards to supporting conceptual modelling through an automated approach:

1. The informal nature of NLRs - NL is incomplete, which leads to problems such as ambiguity and omissions. Under these circumstance, extracting useful information for conceptual modelling becomes a challenging task. Using NLRs as input, the SOM approach implements a guided Q&A process in order to elicit missing information from domain experts.

2. The iterative nature of software development - software development often begins with ill-defined ideas of what the proposed system is to do and clients do not know what they want [Bro87], thus making requirements unstable and in a constant state of evolution. Therefore, a conceptual modelling approach needs
to accommodate the introduction of new information constantly within its structures and create or update the interactions with previously existing information. The SOM approach uses conceptual graphs as data structures. This allows for a comprehensive view of the input NLR, which is flexible and easy to update.

3. Translating NLRs to conceptual models - the approach presented in this paper is an indirect transformation approach, meaning that there is an intermediate model that bridges the gap between NLR and conceptual model. By introducing an intermediate level, the complexity of the transformation increases. Decisions regarding the level of abstraction of this intermediate model will affect the performance of the NLP tools used, the complexity of the output analysis models, as well as the actual usefulness of the approach. For example, the higher the level of abstraction, the larger the size of potential application target, however, utility delivered to modellers is reduced, due to loss of details [Sut02]. Additionally, an indirect transformation should offer the flexibility of mapping its intermediate models to a broad range of modelling languages. The SOM approach addresses this issue through the SOM metamodel. This metamodel allows for the mapping between SOM Patterns and any other modelling language that is define through a metamodel.

4. The evaluation of the output models - the output models obtained after the analysis of an NLR should be consistent and correspond to the information contained in the input NLR. Therefore, the SOM approach developed two methods of model validation. First, a consistency analysis method validates that the output model is semantically consistent (i.e. there are no missing actions or elements, or inconsistent terms used such as synonyms and abbreviations). Second, a paraphrasing method so that domain experts without modelling knowledge can validate the contents of the output models and confirm that the output model corresponds to the informally expressed requirements.

5. The evaluation of the implemented approach - transformation approaches are evaluated based on the types and quality of their output models. Conceptual models obtained during NLR translations are not good or bad, but rather useful or less useful [Fow97], which makes them difficult to assess. One of the purposes of conceptual models is to serve as communication devices between technical and non-technical stakeholders. These stakeholders possess different types of knowledge, namely modelling knowledge and domain knowledge. In
8.2 RESEARCH ACHIEVEMENTS

consequence, an appropriate method of validation should take into account both types of knowledge. In this research, a standard measure of relevance has been adapted from information retrieval based on precision and recall metrics.

8.2 Research Achievements

This research has made the following contributions:

1. A novel model transformation approach (SOM approach) - An approach for the automatic construction of conceptual models to support early stages of software development. The approach uses SOM Patterns to bridge the gap between NLRs and their corresponding conceptual models.

   (a) The creation of Semantic Object Models (SOMs) - these requirements-centred primitive semantic object models were created based on research pertaining to linguistic structures [Cho02, Jac92, Sow84] and cognitive correspondence [Gen83]. SOM Patterns are the central elements of the SOM approach. They capture semantics of the commonly used requirements concepts and their relationships. SOM patterns are motivated by the idea that software requirements statements can be reduced to a small number of recurring concepts.

   (b) The creation of Semantic Object Networks (SONs) - this construct is a knowledge representation network obtained through the composition of SOMi. SOMs and SONs contain the same information, but their perspectives are complementary. While SOMs are defined as being requirement-centred, which allows the representation of each individual requirement and its components, SONs are products of SOMi composition and are concept-centred. This allows the user to understand exactly how concepts within an NLR relate and interact with each other. For example, SONs allow users to view all the actions that a particular agent performs within the system.

2. A systematic validation approach with human in the loop - a standard evaluation test from information retrieval was adapted in order to evaluate the accuracy of the output models (SONs). By using correctness and completeness, the
accuracy of a model is measured by two metrics. First, the correctness metric measures how correct the constructed conceptual model is in representing the original NLR. Second, the completeness metric measures how complete the conceptual model is in covering the concepts contained in the NLR.

3. A set of translation rules and analysis algorithm - these rules are divided into syntactic, semantic, composition, and NL generation rules. Between these four sets of rules, both static and dynamic perspectives of the input NLR are extracted and mapped into the SOMs. Additionally, these rules govern the composition of SOMi into SONs, as well as the analysis of model consistency and NL generation for user validation.

4. The development of a prototype software tool (TRAM) - this prototype was developed using the Java programming language and the Eclipse Modelling Framework (EMF) tool suite. It accepts as input an NLR document and outputs a SON, while at the same time performing further elicitation with the user through a question and answer type of interaction dialogue.

8.3 Limitations and Future Work

The results achieved during the evaluation process support the proposed hypotheses. Nonetheless, limitations of the research need to be made explicit, opening avenues for future work. Limitations can be summarised as follows:

- The SOM approach is a semantic-driven approach confined to the analysis of functional requirements and thus, excludes this work from claiming beyond this limit.

- The current embodiment of TRAM does not address co-referencing and word sense disambiguation, relies on the output of a parser to perform linguistic analysis, and the NL generation is rudimentary.

- Although the evaluation process supports the comprehensiveness claim of the output conceptual models, and the recorded times suggest that the SOM approach implemented through a tool such as TRAM is useful in supporting the requirements engineering process, this claim still needs to be tested on an industrial data set.
There are many possible research directions for the SOM approach. These can be broadly grouped into the following categories: tool support, evaluation in an industrial setting, and new directions:

- The overall TRAM software would benefit from additional user interface development, as well as implementation of co-referencing and word sense disambiguation tools, and parse tree selection for improved NLP.

- The empirical evaluation of TRAM by industrial users would strengthen the claims of the SOM approach. Therefore, an invitation for industry users to engage with TRAM for further evaluation assessment is considered for the future.

- Future work will also consist of investigating new directions such as the study of other theories such as Schank’s conceptual dependency and functional grammar, or the study of Frame Net and its relevance to conceptual modelling, or the possibility of extending the current SOM metamodel to accommodate the coverage of for example intentional aspects such as goal identification.
References


REFERENCES


REFERENCES


REFERENCES


REFERENCES


REFERENCES


REFERENCES


Appendix A

Natural Language Requirements

Examples

A.1 Textual Descriptions of Examples Used in the Practical Evaluation: First Priority Group

Example 1. Library Information System

“A library issues loan items to customers. Each customer is known as a member and is issued a membership card that shows a unique member number. Along with the membership number, other details on a customer must be kept such as a name, address, and date of birth. The library is made up of a number of subject sections. Each section is denoted by a classification mark. A loan item is uniquely identified by a bar code. There are two types of loan items, language tapes, and books. A language tape has a title language (e.g. French), and level (e.g. beginner). A book has a title, and author(s). A customer may borrow up to a maximum of 8 items. An item can be borrowed, reserved or renewed to extend a current loan. When an item is issued the customer’s membership number is scanned via a bar code reader or entered manually. If the membership is still valid and the number of items on loan less than 8, the book bar code is read, either via the bar code reader or entered manually. If the item can be issued (e.g. not reserved) the item is stamped and then issued. The library must support the facility for an item to be searched and for a daily update of records” [Cal94, p. 169].
Example 2. Library Information System II

“The library System is used by the Informatics students and Faculty. The Library contains Books and Journals. Books can be issued to both the Students and Faculty. Journals can only be issued to the Faculty. Books and Journals can only be issued by the Librarian. The deputy-Librarian is in-charge of receiving the returned Books and Journals. The Accountant is responsible for receiving the fine for over due books. Fine is charged only to students and not to the Faculty”.

Example 3. Shoe Company System

“Customers will need to register with the Odd Shoe Company to make orders. On registration, they need to provide name and address, payment details (credit card, etc), shoe sizes, gender, and any special details. To order, customers will select from the shoe range. The system will tell them if the shoes are in stock, or need to be ordered from a supplier. If the shoes are in stock, then the system will tell them how quickly the shoes can be delivered. Customers will want to track their orders online. An order can either be waiting for delivery to the Odd Shoe Company, waiting for dispatch, waiting for credit clearance, or dispatched. Before dispatch, the customer should have an option for cancelling the order. John would like a weekly report, detailing numbers of customers, statistics on shoe sizes by left and right foot, orders, stocks, and cancelled orders. Every month customers will be sent a statement by email, together with a list of special offers. Offers will only be for shoe pairs that are available in suitable sizes for that customer. John wants to keep pictures, prices, stock levels, and sizes of all his shoes in a database. When supplies arrive the database will need updating. When goods are ordered, the stock levels should be deducted. If an order is cancelled, the stock levels should be updated accordingly” [Lun02, p. 377].

Example 4. SalesWeb Application

“Salesperson turns on laptop, brings up the SaleWeb program, and chooses Report Sales Order from menu. Salesperson enters name, employee number, and ID. Sales Order checks to see if name, number and ID
are valid. Salesperson enters customer name and address on sales order form. Salesperson checks customer information to find customer status. CustInfo checks Accounting to determine customer status. Accounting approves customer information and supplies customer credit limit. CustInfo accepts customer entry on Sales Order. Salesperson enters first item being ordered on sales order form. Salesperson enters second item being ordered, etc. When all items have been entered Items ordered are checked to determine availability and to check pricing. Items ordered checks with Inventory to determine availability and to check pricing. Inventory supplies all availability dates (when it can ship), approves prices, adds shipping and taxes, and totals order. Complete order appears on salesperson's screen. Salesperson can print order, check with customer, etc. Salesperson submits the approved Sales Order. Sales Order is submitted to Accounting and Inventory" [HW97, p. 121].

Example 5. Missile Control System

“A launch control center manages a number of missiles. Each missile is controlled by an on-board FMCS unit. The FMCS is initially idle. When the FMCS receives a launch command from its base, if it is idle, the FMCS enters confirmation waiting mode. If the FMCS is in confirmation waiting mode, it receives a launch confirmation command from its base, and the token of the launch confirmation command is equal to the token of the launch command, then the FMCS enters operation mode. When the FMCS receives a launch cancellation command from its base, if it is in confirmation waiting mode, it becomes idle. When the FMCS enters operation mode, it turns on the main engine. A main engine is part of a missile. While the FMCS is in operation mode: every 100 milliseconds, it reads the height from the altimeter, and if the height is greater than 30 meters, it enters cruise mode. If the FMCS is still in operation mode 5 seconds after it entered that mode the FMCS turns off the main engine, and it enters idle mode. During flight, every 20 milliseconds: the FMCS reads the height from the altimeter, it reads the heading from the gyroscope, and it computes appropriate navigation commands, based on the target position. During flight, if the navigation commands are significant, the FMCS sends these commands to the navigation engines. During flight, the FMCS logs
position, height, and heading every 2 seconds. The Launch Control Center sets the target position for the FMCS of its missiles. During flight, the FMCS reads the current position from its GPS device every 5 seconds. A missile has as components: a main engine, a set of navigation engines, a gyroscope, an altimeter, and a GPS device” [AG06].

A.2 Textual Descriptions of Other Examples Used for Evaluation: Second Priority Group

Example 6. Bank Accounts and Transactions System

“The Bank Accounts and Transactions⁴ (BAT for short) system is to be built for the Big Bank Corporation. It must handle clients’ bank accounts and the (standard) services on these accounts: deposit, withdraw, transfer, get balance. The transactions are recorded, because at the end of each month, the system sends out account statements to all clients showing all transactions performed for their accounts during the last period; the system sends the statements to the printer from where a junior clerk posts them. The system is accessed by the bank’s clients only indirectly, i.e., either via a teller, an ATM, or the Internet. All transactions and queries are possible via a teller; all transactions and queries are possible except deposits via an ATM; and all except deposits and withdrawals via the Internet. Opening an account can be performed only via a teller and the Internet; however, if clients open an account via the Internet they must identify themselves with a teller to have their account activated (this is government policy to avoid money laundering, e.g.). Closing an account can only be performed by a teller, and it requires a final statement to be sent out to the client. The Bank offers various account types, which fall into two categories: savings and checking. Savings accounts cannot be overdrawn. There can be a credit limit, subject to agreement by the bank, on checking accounts; a checking account cannot be overdrawn beyond this limit”.

⁴http://softeng.polito.it/courses/02CIX/RE-A2-case-studies/
Example 7. Hotel Management System

“Sogno is a company which owns and manages hotels. It needs to modify its information system. The new system should improve the reservation services in order to get a better level of integration, which would improve the guest service. There are three hotels in the chain. All the hotels are located in the same area. The first hotel, Bolzano, is near the cathedral, in the centre of the town. It is a four-star hotel. It has 15 double rooms and 5 single rooms. Each room contains a bath and a small balcony. The hotel Bolzano has a restaurant, a private car park and a garden. The hotel Koenig is the second hotel owned by Sogno. It is not far from the railway station. It is a big new three-star hotel with 55 rooms. The third hotel, Sirena, is by the sea. It has 30 rooms. All rooms have a balcony. 10 rooms look out onto the sea. The hotel Sirena has a garage, a disco and two restaurants. Guests can make a reservation by calling the hotels or the central office of the company, Sogno. Guests are asked about the arrival-departure time and the type of room. They should leave the name, the telephone number and the way of payment. If there isn’t a vacant room in the hotel called by guests, the receptionist should propose a room in another hotel of the chain. If they cancel the reservation, they lose the deposit” [Mic96, p. 176].

Example 8. Trip Planning

“A Traveller is planning on taking a trip. She decides where to go (her final destination) and when to leave and return. These activities are performed independently on any specific contact the Traveller may have with the Travel Agent (for instance, the trip may be planned using a catalogue or any other means which do not involve connecting to the Travel Agent site). Once she picks her preferred plan, she submits her choice to a Travel Agent by means of her local Web Service software (Order Trip). The Travel Agent evaluates which is the best itinerary to reach the desired destination, based on the Traveller’s criteria such as cheapest price, availability of destination, type of plane or frequent flyer miles. The Travel Agent finds that the best travel plan includes one or more discrete journeys, or which, for the sake of simplicity, will all be operated by the
same Airline. For each leg, the Travel Agent asks the Airline Reservation System to verify the availability of seats (Verify Seats Availability). This availability is constrained by the departing/arrival time information and by the Traveller’s seat choice. For each leg, the Airline Reservation System provides information about the availability of a particular seat. Note that the Travel Agent may have to restart the whole selection in case the Airline would not be able to verify the availability of a given seat. Once the ideal travel plan has been validated, the Travel Agent builds a proposed itinerary for the Traveller to verify. This itinerary might actually not be satisfactory to the Traveller for various reasons. So, the Traveller has the choice of accepting or rejecting the proposed itinerary as well as she may decide not to take the trip at all. In case she rejects the proposed itinerary, she may submit the modifications (change itinerary) and wait for a new proposal from the Travel Agent. In case she decides not to take the trip, she informs the travel agent about her decision (cancel Itinerary) and the process ends. In case she decides to accept the proposed itinerary (reserve Tickets), she will provide the Travel Agent with her Credit Card information in order to properly book the itinerary. The Traveller will, also, provide her contact information for the Airline Reservation System since the Traveller wants to receive an e-Ticket directly from the Airline. Next, the Travel Agent connects with the Airline Reservation System in order to finalize the seat reservation (reserve Seats). The Airline Reservation System electronically reserves the seats associated to each leg; the validity period for such reservation is of one day meaning that if a final confirmation will not be received within one day, tickets will be unreserved and the Travel Agent will be notified. Next, the Travel Agent informs the Traveller about the reservation of the seats. The Traveller can now either finalize the reservation or cancel it. If she confirms the reservation (book Tickets), the Travel Agent asks the Airline Reservation System to finally book the seats (book Seats). The Airline Reservation System books the seats for the chosen itinerary and, then, issues an e-ticket to the Traveller (send Tickets). Finally, the Travel Agent charges the Traveller’s Credit Card with the relevant amount and sends the notification of the charge (send Statement) together with a detailed description of the itinerary to the Traveller”.
Example 9. Order Material

“The first thing we do is request material using a Purchase Request form. Then the Purchasing department either identifies our current supplier for the kind of material requested or sets out to identify potential suppliers. If we have no current supplier for the needed item, Purchasing requests bids from potential suppliers and evaluates their bids to determine the best value. Once a supplier is chosen, Purchasing orders the requested material. Those requesting material must first prepare a Purchase Request. The requester must then obtain the Account Managers approval or that of the designated backup, for the purchase. Purchase Requests submitted for Account Manager approval must include the Account Number for the Project that will fund the purchase. Account Managers or their designated backup, are responsible for, and must approve, all purchases made against their project accounts. After the Account Manager approves the purchase, an authorisation signature may be required. To avoid a potential conflict of interest, the requester cannot be the same individual who approves or authorises the request. Purchase Requests involving Direct projects require an authorisation signature whereas Indirect projects do not. Once all the appropriate signatures are in place, the requester submits the signed Purchase Request to Purchasing. Purchasing then orders the requested material and tracks it as a Purchase Order” [MMP+95, p. 12].

Example 10. Musical Store

“The musical store receives tape requests from customers. The musical store receives new tapes from the Main office. Musical store sends overdue notice to customers. Store assistant takes care of tape requests. Store assistant update the rental list. Store management submits the price changes. Store management submits new tapes. Store administration produces rental reports. Main office sends overdue notices for tapes. Customer request for a tape. Store assistant checks the availability of requested tape. Store assistant searches for the available tape. Store assistant searches for the rental price of available tape. Store assistant checks status of the tape to be returned by customer. Customer can borrow if
there is no delay with return of other tapes. Store assistant records rental by updating the rental list. Store assistant asks customer for his address” [DS09, p. 158].
Appendix B

Related Approaches Output Models

B.1 Example 1. Library Information System

Callan’s model shown in Figure B.1, was manually constructed by Callan [Cal94].

![Class diagram constructed by Callan](image)

Figure B.1: Class diagram constructed by Callan [Cal94, p. 171]

CM-Builder’s model shown in Figure B.2, was automatically generated using the CM-Builder CASE tool.

Gelhausen’s model shown in Figure B.3, was manually constructed by Gelhausen [Gel10].
APPENDIX B. RELATED APPROACHES OUTPUT MODELS

![Class model produced by CM-Builder [HG00]](image)

**Figure B.2**

B.2 Example 2. Library Information System II

**RACE’s model** shown in Figure B.4, was extracted through a prototype tool called RACE (Requirements Analysis and class diagram Extraction) [IA10].

**Shinde et al.’s model** shown in Figure B.5, was extracted by an NL-based system, which automates the building of UML diagrams from free text requirement document.

B.3 Example 3. Online Shoe Company

**Li et al.’s model** shown in Figure B.6, was constructed through an algorithmic approach that employs natural language analysis techniques in the requirements capture process to help structure natural language text.

B.4 Example 4. SalesWeb Application

**Harmon’s model** shown in Figure B.7, was manually constructed by Harmona and Watson [HW97] through the CRC (Classes, Responsibilities, and Collaborators) approach.

B.5 Example 5. Missile Control System

**CIRCE’s model** shown in Figure B.8, was automatically constructed by an environment for the analysis of natural language requirements called CIRCE.
B.5. EXAMPLE 5. MISSILE CONTROL SYSTEM

Figure B.3: Class diagram constructed by Gelhausen [Gel10]
APPENDIX B. RELATED APPROACHES OUTPUT MODELS

Figure B.4: Class diagram extracted by RACE [IA10]

Figure B.5: Class diagram extracted by Shinde et al. system [SBM12]
Figure B.6: Class diagram constructed by Li et al. approach [LDP04]

Figure B.7: Class diagram constructed via CRC approach [HW97, p. 187]
Figure B.8: Class diagram extracted by CIRCE [AG06]