Peter Bellström

Schema Integration

How to Integrate Static and Dynamic Database Schemata
Peter Bellström

Schema Integration

How to Integrate Static and Dynamic Database Schemata
Peter Bellström. *Schema Integration - How to Integrate Static and Dynamic Database Schemata*

**DISSERTATION**

Karlstad University Studies 2010:13
ISSN 1403-8099
ISBN 978-91-7063-299-0

© The Author

Distribution:
Karlstad University
Faculty of Economic Sciences, Communication and IT
Information Systems
651 88 Karlstad
Sweden
+46 54 700 10 00

www.kau.se

Printed at: Universitetstryckeriet, Karlstad 2010
To Villiam and Lisel
Abstract

Schema integration is the task of integrating several local schemata into one global database schema. It is a complex, error-prone and time consuming task. Problems arise in recognizing and resolving problems, such as differences and similarities, between two schemata. Problems also arise in integrating static and dynamic schemata. In this thesis, three research topics are addressed: Maintaining Vocabulary in Schema Integration, Integration of Static Schemata and Integration of Static and Dynamic Schemata, while applying the notation in the Enterprise Modeling approach.

In Maintaining Vocabulary in Schema Integration an analysis of what semantic loss is and why it occurs in schema integration is conducted. Semantic loss is a problem that should be avoided because both concepts and dependencies might be lost. In the thesis, it is argued that concepts and dependencies should be retained as long as possible in the schemata. This should facilitate user involvement since the users’ vocabulary is retained even after resolving similarities and differences between two schemata.

In Integration of Static Schemata two methods are developed. These methods facilitate recognition and resolution of similarities and differences between two conceptual database schemata. By applying the first method, problems between two schemata can be recognized that otherwise could pass unnoticed; by applying the second method, problems can be resolved without causing semantic loss by retaining concepts and dependencies in the schemata.

In Integration of Static and Dynamic Schemata a method on how to integrate static and dynamic schemata is developed. In the method, focus is put on pre- and post-conditions and how to map these to states and state changes in the database. By applying the method, states that are important for the database can be designed and integrated into the conceptual database schema. Also, by applying the method, active database rules can be designed and integrated into the conceptual database schema.
Acknowledgments

A number of people have helped me in the process of writing this thesis. Their input in sharpening and improving my work has been very important and I am very grateful for it.

First I would like to express my gratitude to my colleagues at the Department of Information Systems for reading, discussing and improving my work.

I would especially like to thank my supervisor, Professor Remigijus Gustas, for his support during the years I have been writing this thesis. A special thanks is also due to my co-supervisor, Sten Carlsson, for his encouragement and being a co-author in two of the included papers.

I also thank Lars Jakobsson for his cooperation in writing several of the papers included in this thesis.

A special thanks to Jürgen Vöhringer for introducing me to Klagenfurt and for his cooperation in several of the papers included in this thesis. At the same time I want to thank Professor Heinrich C. Mayr, principal at University of Klagenfurt, for inviting me to the university as a guest researcher.

I would also like to gratefully acknowledge Professor Paul Johannesson for the important contributions made at my final seminar and for being my opponent of my licentiate thesis. Your input has improved and sharpened this thesis!

I am also grateful to Michelle Cryer for proofreading the introductory chapters and to Irina Persson for checking the list of references.

I am also very grateful to Sparbanksstiftelsen Alfas internationella stipendiefond för Karlstads universitet for granting me a scholarship to visit University of Klagenfurt. I would also like to thank The Royal Swedish Academy of Sciences for granting me scholarships to IASTED International Conference on Software Engineering 2008 in Innsbruck, Austria and to International Conference on Information, Process and Knowledge Management 2009 in Cancun, Mexico. I would also like to thank Springer Science and Business Media and ACTA Press for allowing me to reprint already published papers.
I would also like to thank my parents for their support and to thank my sister and my brother-in-law for always encouraging me to finalize the thesis by asking: when are you going to invite us to eat sandwich layer-cake?

I would also thank my friend Fredrik for always triggering and encouraging me to finalize the thesis by addressing me as a “half-doctor”. Now Fredrik I finalized the second half!

Thanks are also sent to my parents-in-law for always showing interest in my work and encouraging me to finalize the thesis. Thanks also to Clas-Göran for showing interest in my work.

Last, but certainly not least, I would like to thank the most important people of my life, my family. Nina, Villiam and Lisel, thanks for the love and support while writing this thesis. This thesis is dedicated to my son Villiam and my daughter Lisel!

Karlstad, April 2010

Peter Bellström
**Contents**

Abstract ................................. i
Acknowledgments ....................... iii
Table of Figures ......................... ix
Table of Tables .......................... xiii

Part I Cover .............................. 1

1 Introduction ............................ 3
   1.1 Research Aim ..................... 5
   1.2 Problems Addressed in the Thesis 6
   1.3 Research Questions and Overall Main Contributions 8
   1.4 A Scandinavian Historical Perspective 9
   1.5 Definitions and Descriptions of Key Concepts 10
   1.6 Disposition of the Thesis ........ 19

2 Research Approach .................... 23
   2.1 Research Environment ............ 23
   2.2 Research Assumptions .......... 24
   2.3 Research Area .................. 25
   2.4 Research Delimitations ......... 26
   2.5 Research Method ................ 27
      2.5.1 Design Science .......... 28
   2.6 Research Process ............... 45
   2.7 Target Group .................. 47

3 Modeling Language and Integration Process 49
   3.1 Modeling Languages .......... 49
      3.1.1 Entity-Relationship Modeling Language 50
      3.1.2 Unified Modeling Language ........ 51
      3.1.3 Klagenfurt Conceptual Pre-Design Model 53
      3.1.4 The Enterprise Modeling Approach ....... 56
      3.1.5 Summary of Modeling Languages and Approaches 60
   3.2 Integration Process .......... 61
      3.2.1 Integration of Static Schemata 61
      3.2.2 Integration of Dynamic Schemata 63
      3.2.3 Integration of Static and Dynamic Schemata 65
      3.2.4 Summary of Integration Approaches 66
# 4 Maintaining Vocabulary in Schema Integration

4.1 Motivation 69  
4.2 Problem, Approach and Solution 70  
4.2.1 Concept Name Compression 71  
4.2.2 Revisiting Concept Name Compression 74  
4.2.3 Semantic Loss 76  
4.3 Contribution 78  
4.4 Contribution and Evaluation in Relation to Design Science 80

# 5 Integration of Static Schemata

5.1 Motivation 83  
5.2 Problem, Approach and Solution 84  
5.2.1 Recognition and Resolution of Homonyms 85  
5.2.2 Inference Rules 88  
5.2.3 Resolution of Homonyms, Synonyms and Hypernyms-Hyponyms 91  
5.2.4 Recognition and Resolution of Similarities and Differences 93  
5.2.5 Semantic Loss in Relation to Recognition and Resolution of Similarities and Differences 96  
5.3 Contribution 99  
5.4 Contribution and Evaluation in Relation to Design Science 108

# 6 Integration of Static and Dynamic Schemata

6.1 Motivation 111  
6.2 Problem, Approach and Solution 112  
6.2.1 A Generic Method to Designing and Integrating Static and Dynamic Database Schemata 113  
6.2.2 Modeling and Integrating States and State Changes to Ensure Database Consistency 118  
6.2.3 Modeling and Integrating Active Database Rules to Ensure Database Consistency 123  
6.2.4 Generic Guidelines for Integration of Dynamic Schemata 126  
6.2.5 A Method for Integrating Conditions (States) and Actions 129  
6.3 Contribution 134  
6.4 Contribution and Evaluation in Relation to Design Science 136

# 7 Conclusions and Future Research

7.1 Conclusions and Implications 139  
7.1.1 Maintaining Vocabulary in Schema Integration 139  
7.1.2 Integration of Static Schemata 140  
7.1.3 Integration of Static and Dynamic Schemata 141  
7.2 Future Research 143  
7.2.1 Future Research – Work in Progress 143  
7.2.2 Future Research – Collaboration in Progress 147  
7.2.3 Future Research – Other Interesting Topics 149  
7.3 Summary of Future Research – Towards New Challenges 151
References 153
Part II Papers 169
Bibliography 325
**Table of Figures**

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>Schema Integration positioned in the Database Design Process</td>
<td>8</td>
</tr>
<tr>
<td>Figure 2</td>
<td>Links and overlaps between research topics</td>
<td>45</td>
</tr>
<tr>
<td>Figure 3</td>
<td>Static schema using the Klagenfurt Conceptual Pre-Design Model (adapted and modified from Bellström et al. (2007 p. 79))</td>
<td>54</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Dynamic schema using the Klagenfurt Conceptual Pre-Design Model (Bellström et al. 2009 p. 27)</td>
<td>55</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Representation of static dependencies in EM (adapted and modified from Gustas &amp; Gustiené (2004))</td>
<td>57</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Representation of dynamic dependencies in EM (adapted and modified from Gustas &amp; Gustiené (2004))</td>
<td>58</td>
</tr>
<tr>
<td>Figure 7</td>
<td>Syntactic elements in EM (adapted and modified from Gustas &amp; Gustiené (2002))</td>
<td>58</td>
</tr>
<tr>
<td>Figure 8</td>
<td>Example of integrated schema including both static and dynamic aspects (adapted and modified from Gustas (2005))</td>
<td>59</td>
</tr>
<tr>
<td>Figure 9</td>
<td>Process of Integration of Static Schemata (adapted and modified from Bellström (2006a p. 20))</td>
<td>62</td>
</tr>
<tr>
<td>Figure 10</td>
<td>Process of Integration of Dynamic Schemata</td>
<td>64</td>
</tr>
<tr>
<td>Figure 11</td>
<td>Method for integrating static and dynamic schemata (adapted and modified from Bellström &amp; Jakobsson (2006 p. 636))</td>
<td>66</td>
</tr>
<tr>
<td>Figure 12</td>
<td>Two synonymic concept names have been recognized</td>
<td>71</td>
</tr>
<tr>
<td>Figure 13</td>
<td>Concept name compression</td>
<td>72</td>
</tr>
<tr>
<td>Figure 14</td>
<td>Homonyms – Size used in the context of TV and Product</td>
<td>75</td>
</tr>
<tr>
<td>Figure 15</td>
<td>Inter-schema hypernym-hyponym property between Order and High Priority Order</td>
<td>75</td>
</tr>
<tr>
<td>Figure 16</td>
<td>Resolution of inter-schema hypernym-hyponym property applying inheritance dependency</td>
<td>76</td>
</tr>
<tr>
<td>Figure 17</td>
<td>Resolution of homonyms applying sharper cardinality, inheritance dependency and new concepts (TV.Size and Product.Size)</td>
<td>76</td>
</tr>
<tr>
<td>Figure 18</td>
<td>Graphical description of Information Loss, Concept Name Compression and Semantic Loss</td>
<td>78</td>
</tr>
<tr>
<td>Figure 19</td>
<td>Resolution of synonyms applying the mutual inheritance dependency</td>
<td>78</td>
</tr>
<tr>
<td>Figure 20</td>
<td>Inheritance and instance-of dependencies in one schema</td>
<td>86</td>
</tr>
<tr>
<td>Figure 21</td>
<td>Recognition of homonyms based on both name and neighborhood matching</td>
<td>86</td>
</tr>
<tr>
<td>Figure 22</td>
<td>Integrated schema with resolved homonyms introducing a new concept and applying inheritance dependency together with sharper cardinality</td>
<td>88</td>
</tr>
<tr>
<td>Figure 23</td>
<td>Recognition of equivalent concepts based on both name and neighborhood matching</td>
<td>89</td>
</tr>
<tr>
<td>Figure 24</td>
<td>Inference rule including both inheritance and instance-dependencies</td>
<td>89</td>
</tr>
<tr>
<td>Figure 25</td>
<td>Integrated schema merged over the TV concept</td>
<td>89</td>
</tr>
</tbody>
</table>
Figure 26. Example of inference rule applying the inheritance and the mutual inheritance dependencies.................................................................90
Figure 27. Inference rule applying inheritance and mutual inheritance dependencies..............................................................................................................90
Figure 28. Example of concept names that might indicate an inter-schema hypernym-hyponym property..............................................................91
Figure 29. Integrated schema with resolved inter-schema hypernym-hyponym property.................................................................................................92
Figure 30. Example of concept names and neighborhoods that might indicate an inter-schema holonym-meronym property ........................................93
Figure 31. Integrated schema with resolved inter-schema holonym-meronym property.................................................................................................94
Figure 32. Identification of synonyms based on both concept name matching and neighborhood matching................................................................95
Figure 33. Example of concept names that might indicate inter-schema properties ..................................................................................................96
Figure 34. Specialization and association.................................................................................................................................................................97
Figure 35. Applying instance-of dependency to resolve the specialization – association problem...................................................................................98
Figure 36. Method for recognition of similarities and differences with focus on equivalent concept names (labels) ..................................................................102
Figure 37. Method for recognition of similarities and differences with focus on similar concept names (labels) .............................................................104
Figure 38. Proposed design and integration process (adapted and modified from Bellström & Jakobsson (2006 p. 636))......................................................114
Figure 39. Schema 1: Sales department (adapted and modified from Bellström & Jakobsson (2006 p. 640))............................................................................115
Figure 40. Schema 2: Warehouse department (adapted and modified from Bellström & Jakobsson (2006 p. 641)).................................................................................116
Figure 41. Integrated static schema (adapted and modified from Bellström & Jakobsson (2006 p. 643)).............................................................................116
Figure 42. Integrated sales and warehouse schema (adapted and modified from Bellström & Jakobsson (2006 p. 644)).................................................................................118
Figure 43. Change of database state illustrated as pre- and post-conditions (adapted and modified from Jakobsson & Bellström (2007 p. 42)) ..........119
Figure 44. Integrated static schema (adapted and modified from Jakobsson & Bellström (2007, p. 40)).......................................................................................................................120
Figure 45. Schema 1: Create order (adapted and modified from Jakobsson & Bellström (2007 p. 43)).................................................................................................121
Figure 46. Schema 2: Update order (adapted and modified from Jakobsson & Bellström (2007 p. 43)).........................................................................................122
Figure 47. Integrated create and update order schema (adapted and modified from Jakobsson & Bellström (2007 p. 43))........................................................................123
Figure 48. Active database rule (adapted and modified from Bellström & Jakobsson (2008 p. 367)).........................................................................................126
Figure 49. Action and pre- and post-conditions (states)..................................................................................................................128
Figure 50. Schema 1: Reserve book......................................................... 131
Figure 51. Schema 2: Cancel Book Reservation.................................... 131
Figure 52. Integrated schema with states Book [Not Reserved] and Book [Reserved] ...................................................................................................................... 132
Figure 53. State and State Changes for Database Consistency (adapted and modified form Jakobsson & Bellström (2007 p. 42)) ........................................... 135
Figure 54. Specialization and Inheritance.............................................. 145
Figure 55. Association............................................................................. 145
Figure 56. Power Type .......................................................................... 146
Figure 57. Instance-of............................................................................. 146
Figure 58. Specialization (inheritance) and instance-of ....................... 147
Table of Tables

Table 1. Summary of defined and described key concepts 11
Table 2. Design Science Research Guidelines (quoted from Hevner et al. (2004 p. 83)) 29
Table 3. Design Science Artifacts (quoted from March & Smith (1995)) 30
Table 4. Design Science Research Methodology (quoted from Peffers et al. (2008)) 30
Table 5. Design Evaluation Methods (quoted from Hevner et al. (2004 p. 86)) 33
Table 6. Overall presentation of research study one (Bellström & Carlsson 2004) 34
Table 7. Overall presentation of research study two (Bellström 2005) 35
Table 8. Overall presentation of research study three (Bellström & Jakobsson 2006) 36
Table 9. Overall presentation of research study four (Bellström 2006b) 37
Table 10. Overall presentation research study five (Bellström & Carlsson 2006) 38
Table 11. Overall presentation of research study six (Bellström et al. 2007) 39
Table 12. Overall presentation of research study seven (Jakobsson & Bellström 2007) 40
Table 13. Overall presentation of research study eight (Bellström 2009) 41
Table 14. Overall presentation of research study nine (Bellström et al. 2008) 42
Table 15. Overall presentation of research study ten (Bellström & Jakobsson 2008) 43
Table 16. Overall presentation of research study eleven (Bellström et al. 2009) 44
Table 17. Resolution methods for homonyms 105
Table 18. Resolution methods for synonyms 106
Table 19. Resolution methods for cyclic generalization and reverse subset relationships 107
Table 20. Resolution methods for the inter-schema hypernym-hyponym property 107
Table 21. Resolution methods for the inter-schema holonym-meronym property 108
Table 22. Guidelines and integration method 133
Part I Cover
1 Introduction

The purpose of this chapter is to present a short introduction and motivation behind the work reported in this thesis. In doing so, the aim of the thesis is presented, followed by the problems addressed and the main contributions. This is followed by a short overview positioning the thesis in a Scandinavian historical modeling perspective. The chapter ends with the definitions and explanations of key concepts used in the thesis and an outline of the disposition of the thesis.

While designing information systems, including databases and software components, it is important that the global graphical description, the integrated schema, illustrates what the users want and clearly communicates the intentions to the designers and users. However, designing and describing the database and information system is a complex task resulting in many and different schemata. The schemata designed during schema design should, during schema integration, be integrated into one global schema since one representation of the database and information system is needed for future development. Moreover, similarities, such as synonyms, and differences, such as homonyms, between two source schemata also have to be recognized and resolved. One of the most quoted definitions of schema integration is given by Batini et al. (1986) where the authors define it as

[...] the activity of integrating the schemas of existing or proposed databases into a global, unified schema. (p. 323).

Differences and similarities between two source schemata arise because the domain of interest can be modeled and represented in many different ways, e.g. on different levels of abstraction or using different properties (Johannesson 1993).

Nevertheless, involving the users during both design and integration is important since several studies (e.g. Rolland 2006) indicate that information systems projects fail due to inadequate or insufficient understanding of what the users are actually trying to communicate to the designers.

The importance of including users in the design process is also emphasized in Molin (2005). The author points out that it is the users that know best what type of support they need in their daily working situation. Therefore, the users are the most suitable persons for requirements elicitation. Involving users in the
design process is also emphasized in the CHAOS report (The Standish Group 1994) where IT executive managers were asked about their opinions regarding important factors that contribute to a successful IT-project. In the first place, executive managers ranked user involvement followed by executive management and finally ending with a clear statement of requirements. In this thesis, the first fact, user involvement, and the third fact, a clear statement of the requirements, are of most importance. The motivation for this is that it is the users that possess the domain knowledge and to end up with an integrated schema, a clear statement of the requirements is needed in the source schemata.

Involving users while conducting schema design should contribute to a more correct collection of schemata compared to no user involvement. At the same time, this gives the designers new problems to solve since each user or user group has his/her own way of naming and representing the domain of interest. This means that while conducting schema integration, differences and similarities between two source schemata have to be recognized and resolved before the schemata are merged into one global conceptual schema. It should be noticed that the term schema integration actually puts focus not only on the process of integrating the schemata, but also on the final product of the process of integrated schema. This is emphasized in Frank (2008) where the author states that the

term integration represents both, a process and its results […] (p. 112).

A conceptual schema might also have many roles and purposes (van Griethuysen 1982; Olivé 2004; Vernadat 1996) and be used for many different purposes (Boman et al. 1997). However, in this thesis, a conceptual schema is mainly treated as a communication “tool” between users and designers (Ambrosio et al. 1997; Hoppenbrouwers et al. 2005b) and as a blueprint, a specification, of the future database and information system (Engels et al. 1992; Parsons 2002).

The method of first designing and later integrating conceptual schemata has been pointed out as important by many researchers. For instance, in Parsons (2002) the author expresses that

[…] database designers should not build a global conceptual schema without first building and verifying local schemas that directly reflect user views. (p. 173).
This can be further explained in several ways. As mentioned by Parsons (2002), local conceptual schemata preserve and highlight differences in how different users view their organization while a global conceptual schema may instead mask these. Nuseibeh et al. (2001) mention that local conceptual schemata may not only prevent premature design decisions but also ensure that all local conceptual schemata are taken into account. Finally, schema integration has been mentioned as an effective – perhaps the most effective technique – for developing (Frank & Eder 1997, 1998) and managing (Mannino 2007) large database schemata. Using a semantic model during conceptual design also simplifies integration of conceptual schemata (Ekenberg & Johannesson 1995).

To summarize, even though a lot of research has been put into schema design and schema integration, many research questions, such as how to integrate static and dynamic schemata, are still not resolved. The need to resolve these questions is the basis for the research and the results reported in this thesis. Wand and Webber (2002) comment on conceptual modeling in general and point out that

> Despite the importance of conceptual modeling, anecdotal and research evidence suggest that it is not done well. Practitioners report that conceptual modeling is difficult and that it often falls into disuse within their organizations. (p. 364).

Roddick and de Vries (2006) also state in connection to schema integration that

> Despite the research to date, schema integration, evolution and versioning are far from being solved as evidence by the almost total lack of functionality in commercial DBMS. (p. 213).

Even though the quotations above clearly state that research is needed in both schema design and schema integration, the work reported in this thesis is mainly focused on the latter: *schema integration*. More precisely, this means integration of schemata created during conceptual design while using a graphical notation provided by the chosen modeling language.

### 1.1 Research Aim

While conducting schema integration, we not only need methods and approaches, but also concrete guidelines on how to recognize and resolve similarities and differences between two source schemata in the integration
Therefore, the overall research aim of the work reported in this thesis is to:

formulate, develop and propose methods that facilitate schema integration while applying the notation given in the Enterprise Modeling approach.

In other words, in this thesis I have dealt not only with the integration of static schemata and the integration of dynamic schemata, but also with the integration of static and dynamic schemata resulting in a global schema.

1.2 Problems Addressed in the Thesis

The main problem studied in this thesis is schema integration. Schema integration is the process of merging several local schemata into one global schema. However, conducting schema integration results in several problems that need to be recognized and resolved before the schemata are merged into one schema. In this thesis, several such problems have been studied and divided into three research topics.

In the first research topic, *Maintaining Vocabulary in Schema Integration*, (studied in Bellström & Carlsson (2004, 2006) and Bellström (2009)), the problem of semantic loss and how to avoid it in schema integration are studied. More precisely this means studying what semantic loss is and how differences and similarities between two source schemata could be resolved without impoverishing the vocabulary used in the source schemata. The notation given in the Enterprise Modeling approach has been applied to illustrate and demonstrate the addressed problems and solutions. In the first research topic, focus is on integration of static schemata designed in conceptual database design (see Figure 1).

In the second research topic, *Integration of Static Schemata*, (studied in Bellström (2005, 2006b, 2009), Bellström & Carlsson (2006) and Bellström et al. (2007)), the problem of recognition and resolution of differences and similarities between two source schemata by applying the notation given in the Enterprise Modeling approach are studied. Focus is again on integration of static schemata designed in conceptual database design (see Figure 1).

In the third and last research topic, *Integration of Static and Dynamic Schemata*, (studied in Bellström & Jakobsson (2006, 2008), Jakobsson & Bellström (2007)
and Bellström et al. (2008, 2009)), two problems in conceptual and database design and integration are studied, applying the notation given in the Enterprise Modeling approach. The first problem deals with how to develop a generic modeling method which includes both modeling and integration of static and dynamic schemata. The second problem deals with issues on how to, on a conceptual level, using pre- and post- conditions, describe and illustrate legal states and state changes in the future implemented database. This also includes how to, on a conceptual level, describe and illustrate active database rules.

In Figure 1, schema integration is illustrated and positioned, by gradual refinement, in the database design process. The arrows pointing upwards should be interpreted as aggregation, meaning, schema design and schema integration are part-of conceptual database design (see for instance Parsons (2002) and Mannino (2007)). Furthermore, pre-integration, comparison of the schemata, conforming the schemata and merging and restructuring are part-of schema integration (see Batini et al. (1986)).

Figure 1 also highlights that the work presented in this thesis focuses on conceptual design. Even though part of the work in this thesis touches upon schema design (see for instance Bellström & Jakobsson (2006)) the main contributions are given in the research area of conceptual schema integration. It should also be noted that the work presented in this thesis also has contributions to integration of logical schemata (see for instance Bellström & Jakobsson (2006, 2008) and Jakobsson & Bellström (2007)).
1.3 Research Questions and Overall Main Contributions

Each of the research topics given in this thesis has its own contributions to the research area of schema integration. However, from a broader perspective, six main contributions are identified, two for each research topic.

In this thesis three research topics are considered as all having their own research focus. Each research topic has its own research questions and contributions (see chapter 4-6) and can be read both as individual chapters and as one contribution the research area of schema integration.

Research topic one, *Maintaining Vocabulary in Schema Integration*, has two research questions:

- *What is semantic loss in connection with schema integration?*
- *How can semantic loss be avoided in schema integration?*

The main contributions of the first research topic are the analysis of what semantic loss is and why it occurs and the demonstrated resolution methods for differences and similarities between two source schemata.
Research topic two, *Integration of Static Schemata*, has one research question:

- **How can similarities and differences between two source schemata be recognized and resolved in schema integration?**

The main contributions of the second research topic are the methods on how to recognize and resolve, without causing semantic loss, similarities and differences between two source schemata.

Finally, research topic three, *Integration of Static and Dynamic Schemata*, has two research questions:

- **How can static and dynamic schemata be integrated?**
- **How can legal states and state changes in the future database be illustrated and integrated during conceptual database design?**

The main contributions of the third research topic are the method on how to integrate static and dynamic schemata and the method on how to illustrate, describe and integrate legal states and state changes in the database using only pre- and post-conditions.

### 1.4 A Scandinavian Historical Perspective

Although the paper in which Chen first described the Entity-Relationship modeling language (Chen 1976) is one of the most important and most known contributions to conceptual database design, it was not the first reported and published work in the field. In Scandinavia, several researchers, had already conducted and published research results. Two such research streams that have roots influencing the work in this thesis are formal approaches and the infological approach (Iivari & Lyytinen 1998). Both research streams originate from Langefors work reported in *Theoretical Analysis of Information Systems* (Langefors 1978) first published in 1966. Following the infological approach, Bo Sundgren (1973) proposed a formalism for conceptual modeling (Iivari & Lyytinen 1998). Sundgren (1973) also discussed and described the concept of “database schema” for this work as a very important concept. Sundgren (1973) described the concept as follows:
a schema is a statement of a set of (references to) object types, attributes, object relations, generation rules, constellation types, internal and external definitions, etc. (p. 127)

Janis Bubenko Jr. (1973) also followed Langefors work; however he focused on the formal approaches research stream doing research on analysis and design of data processing systems. Bubenko Jr. also contributed to conceptual modeling research focusing on database theory, by combining ideas from systems design, knowledge representation and databases (Iivari & Lytinen 1998).

A third researcher who also contributed to the research stream of formal approaches is Arne Solvberg. According to Iivari & Lytinen (1998), Sølvberg focused on how to integrate behavioral modeling with static structural modeling using Petri nets (see Sølvberg & Kung 1986).

As we can see, several Scandinavian researchers have during the years contributed to the research field of conceptual database design which has in turn influenced the work in this thesis.

1.5 Definitions and Descriptions of Key Concepts

The following section includes definitions, descriptions and clarifications of key concepts used throughout the thesis. Although some of the descriptions given below will be addressed and described later in the thesis, I have chosen to collect them all in this section. The motivation for doing so is to have one section where all the key concepts of the thesis are collected and described. The first key concepts are the ones used in the research questions and the key concepts that follow are used frequently in the thesis and are therefore important for the thesis. The first three key concepts are commonly used throughout the thesis and therefore need to be defined first. These are conceptual schema, conceptual view and conceptual database design. The definitions given below originate from the domain of database research. This is natural since schema integration as such also originates from the domain of database research. Nevertheless, the definitions are still applicable on conceptual design in general. A summary of the key concepts addressed in this section is given in Table 1.
Table 1. Summary of defined and described key concepts

<table>
<thead>
<tr>
<th>DEFINED AND DESCRIBED KEY CONCEPT</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Conceptual schema</td>
<td></td>
</tr>
<tr>
<td>Conceptual view</td>
<td></td>
</tr>
<tr>
<td>Conceptual database design</td>
<td></td>
</tr>
<tr>
<td>Schema integration</td>
<td></td>
</tr>
<tr>
<td>Semantic loss</td>
<td></td>
</tr>
<tr>
<td>Minimality</td>
<td></td>
</tr>
<tr>
<td>Information preserving transformation</td>
<td></td>
</tr>
<tr>
<td>Constraint preserving transformation</td>
<td></td>
</tr>
<tr>
<td>Information lost</td>
<td></td>
</tr>
<tr>
<td>Concept name compression</td>
<td></td>
</tr>
<tr>
<td>Conflict</td>
<td></td>
</tr>
<tr>
<td>Homonym</td>
<td></td>
</tr>
<tr>
<td>Synonym</td>
<td></td>
</tr>
<tr>
<td>Inter-schema property</td>
<td></td>
</tr>
<tr>
<td>Hypernym-hyponym</td>
<td></td>
</tr>
<tr>
<td>Holonym-meronym</td>
<td></td>
</tr>
<tr>
<td>Linguistic conflict</td>
<td></td>
</tr>
<tr>
<td>Static schema</td>
<td></td>
</tr>
<tr>
<td>Dynamic schema</td>
<td></td>
</tr>
<tr>
<td>State and state change</td>
<td></td>
</tr>
<tr>
<td>Transaction</td>
<td></td>
</tr>
<tr>
<td>Database consistency</td>
<td></td>
</tr>
<tr>
<td>Pre- and post-condition</td>
<td></td>
</tr>
<tr>
<td>Antonym</td>
<td></td>
</tr>
<tr>
<td>Binary strategy</td>
<td></td>
</tr>
<tr>
<td>Power type</td>
<td></td>
</tr>
</tbody>
</table>

Regarding the first key concept, **conceptual schema**, I have chosen to quote Batini et al. (1992) where the authors stated that a

conceptual schema is a high-level description of the structure of the database, independent of the particular DBMS software that will be used to implement the database. (p. 6)

The closely related concept of view was described by Navathe et al. (1986) as the

perception of users about what a proposed database (or an ideal database) should contain. (p. 50).
Having these two definitions in mind a **conceptual schema** is in this thesis defined as:

*a high-level description of the structure and/or behavior of the information system independent of particular tools and platforms used to implement the information system.*

Finally, in this thesis, **conceptual database design** (cf. Bellström 2006a p. 6) is described and defined as:

*the process of constructing a high-level and implementation independent description, a conceptual schema, of the structure and/or behavior of the database, based on the information used in the enterprise.*

The key concepts used in the research questions for research topic one, *Maintaining Vocabulary in Schema Integration*, are schema integration and semantic loss. The concept of **schema integration** has been given many definitions and explanations in the literature such as:

[…] the activity of integrating the schemas of existing or proposed databases into a global, unified schema. (Batini et al. 1986 p. 323)

In the context of information systems development, the integration of a number of local schemas into a global one is called *view integration*. A similar activity, called *database integration*, occurs in distributed database design, where a set of schemas for already existing information systems is merged into a single global schema. (Boman et al. 1997 p. 150)

[…] the process of merging several conceptual schemas into a *global conceptual schema* that represents all the requirements of the application. (Batini et al. 1992 p. 119)

As illustrated above, all three definitions originate from the domain of conceptual database design research. In that domain, schema integration is often referred to as view integration (see for instance the definition given by Boman et al. (1997)). However, the main application is still to integrate two conceptual source schemata into one global conceptual schema. In this thesis, schema integration is used whenever two conceptual source schemata e.g. two static database schemata, two dynamic schemata, or one static schema and one dynamic schema are integrated into one global conceptual schema.
Semantic loss is a complex concept and today there is no clear definition or description. One explanation for this could be that schema integration is conducted on different levels of abstraction using different modeling languages and approaches resulting in different definitions and explanations of the problem. Another explanation could be that semantic loss and information loss are sometimes used synonymously and therefore often mixed up. Nevertheless, in this thesis, semantic loss in connection with schema integration is defined and described as follows:

A problem that occurs if one or several concept(s), including their names, and/or dependency(ies) describing the meaning of a concept are lost in schema integration.

(Modified from Bellström 2009 p. 965)

More precisely, this means that when one resolves similarities and differences between two source schemata, renaming the concepts would cause semantic loss. This is the case since one concept name is lost which, for one or several users, might mean and refer to something important in the schema. The concept of semantic loss is closely related to minimality, information preserving transformation, constraint preserving transformation, and information loss and concept name compression.

Minimality is used in the literature in connection not only with information loss but also semantic loss. Batini et al. (1992) stated that a schema is minimal when every aspect of the requirements appears only once in the schema. We can also say that a schema is minimal if no concept can be deleted from the schema without losing some information. (p. 140)

The definition given by Batini et al. (1992) refers to a conceptual schema close to implementation. Two related concepts are information preserving transformation (cf. information loss) and constraint preserving transformation (cf. constraint loss). Lee and Ling (2003) describe these as follows:

[...] a transformation is information preserving if any database instance structured according to the original schema can be losslessly converted into a database instance according to the transformed schema, and vice versa [...].

[...] a transformation is constraint preserving if the constraints expressed in the original schema can also be expressed in the transformed schema. (p. 238)
Similar to the definition of minimality given by Batini et al. (1992), the description of both information preserving transformation and constraint preserving transformation refers to a conceptual schema very close to implementation. Nevertheless, in this thesis information loss in connection with schema integration is defined and described as:

*a problem that occurs if one or several concepts and/or dependencies that make it possible to store a real-world fact in the future database are lost in schema integration.*

The last key concept in connection with semantic loss is concept name compression defined by Bellström (2006a) as:

> a state which may occur if several concept names are merged (compressed) into one concept name, for example when choosing one of the concept names to represent a concept when trying to resolve a synonym conflict. (p. 46)

The key concepts used in the research question in research topic two, *Integration of Static Schemata, are similarities and differences between two source schemata.* In this thesis, similarities not only mean ‘synonyms’ but also inter-schema properties such as a hypernym-hyponym. Differences between two source schemata are, for instance, homonyms. In the literature, synonyms and homonyms are often referred to as name conflicts (e.g. Batini et al. 1986; Lee & Ling 2003; Spaccapietra & Parent 1994). However, the concept of conflict has been given different definitions, depending on what the author(s) include into the concept. In addition, several conflict classifications have also been proposed such as the ones given in Batini et al. (1986) and Spaccapietra and Parent (1991) later adapted by Dupont (1994). A more detailed discussion and description of conflict classifications is given in Bellström (2006a). One explanation of the concept of conflict is given by Batini et al. (1986) were the authors state that:

> A *conflict* between two representations R₁ and R₂ of the same concept is every situation that gives rise to the representations R₁ and R₂ not being identical. (p. 335).
In the explanation given by Batini et al. (1986) R₁ and R₂ are used as abbreviations for representation one and representation two. In this thesis linguistic conflict, e.g. homonyms and synonyms have been given much focus and these are therefore explained and defined in the next paragraph.

When dealing with the recognition and resolution of not only similarities but also differences between two source schemata, so-called linguistic conflicts often appear. Two such conflicts are homonyms and synonyms, also known as name conflicts. Homonyms occur if one name is used for several concepts with different meanings while synonyms occur if two or more names are used for one concept with the same meaning.

Although the concept of inter-schema property often occurs while recognizing and resolving similarities and differences between two source schemata, it is not really a conflict but signifies that two concepts have something in common. More precisely, this means that an inter-schema property occurs when two views have certain constraints in common (Johannesson 1993). Examples of inter-schema properties are if one concept is-a another concept or one concept is-part-of another concept. In other words, an inter-schema property occurs when two concepts have a specific dependency between each other. The concept of inter-schema property is closely related to hypernym, hyponym, holonym, meronym and linguistic conflict.

As mentioned above, an inter-schema property is not really a conflict but instead indicates a certain dependency between two concepts. Two such dependencies are hypernym-hyponym and holonym-meronym. A hypernym-hyponym inter-schema property occurs if one concept is recognized as a specialization of another concept which at the same time is recognized as a generalization of the other concept. This is also known as inheritance (is-a) and generalization-specialization. A holonym-meronym inter-schema property occurs if one concept is recognized as a part-of another concept which at the same time is recognized as aggregated or composed of the other(s) concept(s). This is also known as aggregation and composition.

The concept of linguistic conflict is also used in the thesis. Strictly speaking, a linguistic conflict is just one way to assemble the problems that might appear with naming concepts. More precisely, this means that a linguistic conflict could
be one of the following: a homonym, a synonym, a hypernym, a hyponym, a holonym or a meronym.

The key concepts used in the research questions for research topic three, *Integration of Static and Dynamic Schemata*, are *static schema*, *dynamic schema*, *state* and *state change* and finally *pre-* and *post-condition*.

In this thesis, a **static schema** describes, on a conceptual level, not only the static structure of the future database (concepts and dependencies) but also the data structure that is needed and processed by the application accessing the database. The definition given by Batini et al. (1992) is therefore adopted in which the authors stated that a

**conceptual schema** is a high-level description of the structure of the database, independent of the particular DBMS software that will be used to implement the database. (p. 6)

On the other hand, a **dynamic schema** is not the same as a static schema but often parts of the static schemata are included in the dynamic schemata. Nevertheless, both are still described and illustrated on a conceptual level with the extension of including some syntactic elements. Including both static and dynamic semantic dependencies as well as syntactic elements into the schema, we are able to describe who is using what data. More precisely, this means that we are able to illustrate the static concepts needed, including the dependencies between them, the interaction, including flows, states and state changes between two or more actors such as databases, database management systems and software components. It is important to remember that in some of the papers in Part II (Bellström & Jakobsson 2006; 2008; Jakobsson & Bellström 2007), software components are part of the illustrated schemata. However, this thesis puts focus on schema integration in conceptual database design. Consequently, definitions and related topics on software components are excluded. The reader should refer to Christiansson and Christiansson (2006) and Jakobsson (2009) in which the authors discuss and explain software components in more depth.

The concept of **state and state change** also originates from the domain of database research. More precisely, it originates from the research on database consistency which means that the database fulfills all constraints that are
defined on a specific database. Hearder and Reuter (1983) explain database consistency as follows:

> [...] a database is consistent if and only if it contains the results of successful transactions. (p. 291)

The definition and explanation given above has an explicit focus on the legal states and an implicit focus on state changes. A slightly different explanation and definition, with focus on the movement from one consistent state to another consistent state and as well as the result, is given by Connolly and Begg (2010) who writes that a transaction must transform the database from one consistent state to another consistent state. (p. 573)

Before moving on to pre- and post-conditions definitions of transaction and consistency are needed. A transaction is a logical unit of work which should execute all or none of the database changes asked by an application. Connolly and Begg (2010) define a transaction as an action, or series of actions, carried out by a single user or application program, that reads or updates the contents of the database. (p. 570)

In this thesis database consistency is defined and described as follows:

> a database is consistent if none of the constraints defined on the database are violated.

The last important key concept in the last research topic is pre- and post-conditions. As discussed and described in the former section, a condition is something that must hold before or after an action has taking place. This means that a pre-condition is something that must hold prior to an action and a post-condition is something that must hold after an action. It should be noted that a condition can be both a pre- and post-condition since one condition can be the result (post) of one action while the same condition is a prior condition (pre) for the following action.

The last three key concepts defined and described in this section are antonym, binary strategies and power type. Buchholz et al. (1997) describes antonyms as follows:
Antonyms are words of contrastive or opposite meaning. (p. 40).

Antonyms might be useful while checking that the integrated schema is complete. For instance, in the library domain books are mostly borrowed and returned by the customers. If one of these schemata is not described then we have an incomplete set of schemata to be integrated. Antonyms might then be used since they indicate, as defined by Buchholz et al (1997),

[...] contrastive or opposite meaning. (p. 40).

In this thesis, binary strategies relate to the number of source schemata being compared and integrated. In this thesis, it is always assumed that there are two source schemata being compared and integrated in each iteration of the integration process. Batini et al. (1986) explain the advantage of using binary strategies as follows:

Binary strategies allow the integration of two schemas at a time. [...] The advantage of binary strategies is in terms of simplifying the activities of comparison and conforming at each integration step. (p. 343)

This is reinforced by Batini et al. (1992) who state that

[...] only one pair of schemas be considered at a time [...] (p. 122)

This means that the main advantage of using a binary strategy is the simplicity compared to other strategies. Finally, the last key concept that needs to be described is power type. A power type is a problem that needs to be resolved during conceptual database design. Martin and Odell (1998) describe a power type as

a type whose instances are subtypes of another type. (p. 252)

A similar description of the problem is given in the UML Superstructure Specification, v. 2.2 (Object Management Group 2009) as follows

a power type is a class whose instances are subclasses. (p. 55).

In conceptual database design, power types are important to recognize since these need to be resolved and marked out in a proper manner, otherwise it
could result not only in semantic loss but also storing redundant data in the future database

1.6 Disposition of the Thesis

This thesis is divided into two parts. Part I is divided into seven introductory chapters. Part II is divided into eleven chapters, where each chapter contains one of the papers. The numbers inside the parentheses in the next section indicate the chapter and section where the topic is described and discussed.

Part I is organized as follows: after the introduction, the research approach is presented in chapter two. Chapter two includes research environment (2.1), research assumptions (2.2), research area (2.3), research delimitations (2.4), research method (2.5), research process (2.6) and finally the target group (2.7).

In chapter three, the modeling language and the integration process are described. For that purpose, chapter three includes a section about modeling languages (3.1) in which an overview of three modeling languages, the Entity-Relationship modeling language (3.1.1), the Unified Modeling Language (3.1.2), and the Klagenfurt Conceptual Pre-Design Model (3.1.3), are described. In addition, a more detailed discussion and description of the Karlstad Enterprise Modeling approach (3.1.4) is also presented. The modeling language section ends with a summary of the described modeling languages and approaches (3.1.5). Chapter three also includes a section about the integration process (3.2), divided into integration of static schemata (3.2.1), integration of dynamic schemata (3.2.2) and integration of static and dynamic schemata (3.2.3). The discussion and description of the integration process ends with a summary of the addressed integration approaches/processes.

Chapters four to six include a summary of each research topic divided into motivation (4.1, 5.1 and 6.1), problem, approach and solution (4.2, 5.2 and 6.2), contribution (4.3, 5.3 and 6.3) and contribution and evaluation in relation to design science (4.4, 5.4 and 6.4).

Finally, chapter seven includes the conclusions and implications of the research results presented in the thesis (7.1) and proposals regarding future research in the area of schema design and schema integration (7.2).
As mentioned above, Part II contains the eleven source papers in order of presentation and not the order of publication. The papers included in Part II are:


Finally, a comment regarding the use of language and the formatting of papers in this thesis is required. The language in Part I, chapters one to seven, has been proofread, which part II has not. Therefore, the language used can vary somewhat between Part I and Part II. The formatting of each included paper of this thesis is the version sent for printing. However, conference organizers
and/or publishers might have made changes to the formatting. Therefore, the formatting could vary some when comparing the printed version of the paper in this thesis with the printed version in the proceedings.
2 Research Approach

The purpose of this chapter is to describe and discuss the applied research approach. This chapter is therefore divided into seven sections starting with research environment, moving on to research assumptions, research area, research delimitations, research method, research process, and ending with target group.

2.1 Research Environment

In Information Systems at Karlstad University there are four research groups:

- Business driven IT design
- User-oriented interaction design
- Projects as work form
- Enterprise and systems architecture design

The research reported in this thesis has been conducted within the fourth and last of the four research groups: Enterprise and systems architecture design. The research leader is Professor Remigijus Gustas who is also one of the founders behind the modeling language and approach applied in this thesis. As the name of the research group indicates, research is carried out in connection with enterprise and systems architecture design, and in this thesis the application is schema integration. At Karlstad University, Information Systems is both a subject and a department and our discipline can be described as a

[...] multidisciplinary subject with elements of technical, economic and pedagogical aspects.  

The content of this thesis is placed in the first of these: technical aspect.

In this thesis, I have applied the notation given in the Enterprise Modeling approach. There are two ways to describe the Enterprise Modeling approach: as a generalization and extension of system analysis (Gustas & Gustiené 2004) and as a modeling approach and language for modeling and integrating business processes (Vernadat 1996). Finally, by applying the notation given in the Enterprise Modeling approach, it is possible to describe and illustrate not only pragmatic and semantic aspects, but also syntactic aspects, logical elements, of the future information system using one schema type (Gustas & Jakobsson 2004).
Apart from the Enterprise and systems architecture design research group, I have been collaborating with another research group: Research Group Application Engineering at Institute for Applied Informatics University of Klagenfurt Austria. The research leader is Professor Heinrich Mayr who is also vice-chancellor for the University. However, the larger part of the work reported in this thesis has been conducted within the Enterprise and systems architecture design research group.

2.2 Research Assumptions

Designing an information system including both the static and the dynamic aspects is a complex task. Following the classical methodology for designing databases, design starts with conceptual design, moving on to logical design and ending with physical design (Connolly & Begg 2010). This thesis focuses on the first of these three: conceptual design, with the extension of including some syntactic (logical) elements. In this thesis, conceptual design is also seen as a task divided into two sub tasks: schema design and schema integration. During schema design, schemata for each user and/or group of users are defined and during schema integration, the schemata are integrated into one global conceptual schema. The work reported in this thesis focuses on the second part of conceptual design: schema integration. The thesis therefore has the following five research assumptions:

1. An information system, including the databases, the software components and so forth, is defined and designed in autonomous teams made up of both users and designers.

2. Each autonomous team produces one or several schemata, sometimes referred to as views, illustrating some part of the future information system.

3. Each autonomous team has its own focus, its own use of vocabulary and its own way of designing the schemata while applying the primitives of a chosen modeling language.

4. Schemata produced by the autonomous teams overlap and between the schemata a mapping often exists.

5. Therefore the produced schemata may be integrated into a global schema by applying a schema integration process and/or method.
The motivation behind these assumptions is that the user knows best what type of support he/she needs in his/her daily working situation. The assumptions also take into account that defining and developing an information system is a complex, time-consuming and error-prone task. We therefore need to decompose the design procedure into smaller parts (cf. Langevors’ work on unperceivable systems described in Langevors (1995)) and then integrate these parts during schema integration. In doing so, we are able to address two of the most important factors that, according to the CHAOS report (Standish Group 1994), contribute to a successful IT-project: user involvement and a clear statement of requirements.

2.3 Research Area

In the schema integration research area, researchers study the problem of how to integrate several schemata of the domain and/or enterprise in focus. While integrating the schemata, several problems have to be addressed, since dealing with schemata intended to be merged means dealing with different representations of the same database and/or information system. More precisely, this means that different schemata might be illustrated and designed in different ways, such as using different concepts and concept names to illustrate the same part of the information system. In an extreme situation, the schemata might even be described using different modeling approaches and languages. Therefore, two main problems that have to be addressed in the schema integration research area are how to recognize and resolve similarities and differences between two source schemata.

One method for describing the history and the progress of the research conducted within the schema integration research area is to divide it into three parts (see Bellström 2006a) as follows:

- Manual approaches to schema integration
- Formal approaches to schema integration
- Semi-automatic approaches to schema integration

Manual approach means that all tasks are done manually. A formal approach means that some type of formal language is used. Finally, a semi-automatic approach means that at least one computer application is used. The work and contributions presented in this thesis are placed within the first and last of the three approaches. In other words, this thesis contributes to both manual
schema integration and semi-automatic schema integration. However, it should be noted the approaches are not mutually exclusive but rather complementary. For instance, one approach might use a formal and manual approach while another might use a formal and semi-automatic approach. A third combination might be to first apply a computer application, a semi-automatic approach, to recognize and propose resolution methods for similarities and differences between two source schemata followed by manually resolving the recognized similarities and differences. However, this last scenario is not truly a manual approach since using a manual approach means that all tasks are conducted manually.

2.4 Research Delimitations

Although the work presented in thesis includes several contributions on how to perform and support schema integration, both the static and the dynamic constituents, by applying the notation given in the Enterprise Modeling approach, there are still issues that are not taken into consideration. In this thesis, these research delimitations can be divided into two parts. Part one concerns delimitations in connection with the chosen modeling language. Part two concerns delimitations in connection with the three schema integration approaches (manual, formal and semi-automatic) described in the former section. In the rest of this section, the two research delimitations are first described and discussed. This is followed by a description and discussion regarding the delimitations and focus for each of the three research topics given in the thesis.

The first delimitation concerns the chosen modeling language and approach. By applying the Enterprise Modeling approach, it is possible to represent, illustrate and describe not only pragmatic aspects and semantic aspects (both static and dynamic) but also syntactic elements of the domain and/or enterprise in focus in one schema. However, in this thesis the pragmatic dependencies used to define and illustrate goals, problems and opportunities are not taken into account. The motivation for this is to focus on the semantic aspects (static and dynamic).

The second delimitation concerns the integration approaches that are researched in the schema integration community. Pure formal and automatic approaches are not taken into account. Instead, focus is put on manual approaches and semi-automatic approaches. In connection to semi-automatic approaches, it is
important to point out that users should be engaged in the integration process since they possess the domain knowledge. A computer application should only be viewed as a tool that facilitates the integration process, and it is still up to the users to accept or decline the integration proposals given by the computer application.

In both research topics *Maintaining Vocabulary in Schema Integration* and *Integration of Static Schemata* one more narrow and detailed delimitation is of importance. In both research topics, only the static semantic aspects of the Enterprise Modeling approach are taken into account. The aim is to resolve problems, approaches and solutions in schema integration during conceptual database design.

In the third research topic, *Integration of Static and Dynamic Schemata*, there is another detailed delimitation of note. The delimitation again concerns the chosen modeling language the Enterprise Modeling approach. However, in this topic both the static and dynamic semantic aspects are included, as well as a few syntactic elements. For instance, syntactic elements for databases and software components are used to illustrate which actor is initiating a specific action.

### 2.5 Research Method

In the book *The Sciences of the Artificial*, Herbert Simon (1969) distinguishes between “natural sciences” and “artificial sciences” stating that the

natural sciences are concerned with how things are (p. 58)

while design

is concerned with how things ought to be, with devising artifacts to attain goals. (p. 59)

While conducting research in Information Systems, a discipline that focuses on problem solving

[...] at the intersection of information technology (IT) and organizations. (Peffers et al. 2008 p. 46)

it is important to address questions and issues inside both the “natural” and the “artificial” sciences. The need to address issues in more than one science is also
pointed out by Hevner et al. (2004), where the authors distinguish between the “behavioral science paradigm” and the “design science paradigm”. The big difference between these two paradigms is that behavioral science

[...] seeks to find ‘what is true’ (Hevner et al. 2004 p. 98)

while design science

[...] seeks to create ‘what is effective’ (Hevner et al. 2004 p. 98).

While studying research questions in the behavioral science paradigm (Simon (1969) uses natural sciences), often more traditional empirical research is conducted (see for instance Kaplan & Duchon (1988) and Wynkoop & Russo (1997)). Regarding the design science paradigm, an increased interest was raised in the early 1990s among Information Systems researchers (Peffers et al. 2008). One explanation for this increase might be that design science could play an important role while trying to solve the relevance problem, a dilemma that has been troublesome for Information Systems research for many years (Iivari 2007; Lee 2000). The research reported in this thesis addresses issues in relation to design science and is therefore also classified as design science research. This is even more motivated by the fact that the work is more theoretical than empirical and the aim has been to develop a method for schema integration while applying the notation of the Enterprise Modeling approach (cf. Hevner et al. 2004; March & Smith 1995).

2.5.1 Design Science

Design science is not a new research method, and as a paradigm, it has been used for a long time in Computer Science, Software Engineering and Information Systems aiming to produce artifacts (Iivari 2007). However, since the early 1990s, design science has caught more interest from researchers within the Information Systems discipline (Peffers et al. 2008). In design science, the artifact is in focus. Hevner et al (2004) express this as follows:

The result of design-science research in IS is, by definition, a purposeful IT artifact created to address an important organizational problem. (p. 82)

In the above quotation, IS is used as an abbreviation for Information Systems and artifacts can be classified as constructs, models, methods and instantiations (Hevner et al. 2004; March & Smith 1995). In March & Smith (1995) the
authors proposed a research framework in information technology, where research outputs (constructs, models, methods & instantiations) were separated from research activities (build, evaluate, theorize & justify). This distinction clearly showed that in design science, the researcher builds and evaluates artifacts classified as constructs, models, methods and instantiations. The work by March & Smith (1995) was later refined in Hevner et al. (2004). They argued that to reach a good understanding of the results of design science research, researchers should follow seven guidelines. These guidelines are described in Table 2 quoting Hevner et al. (2004).

The first guideline is named Design as an Artifact and construct, model, method and instantiation are again addressed. Each of these is important while conducting design science research because it is artifacts that constitute the actual result and contribution of the research. As a result, each type of artifact is explained in Table 3. The explanations are quoted from March & Smith (1995).

<table>
<thead>
<tr>
<th>GUIDELINE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Guideline 1: Design as an Artifact</td>
<td>Design-science research must produce a viable artifact in the form of a construct, a model, a method, or an instantiation.</td>
</tr>
<tr>
<td>Guideline 2: Problem Relevance</td>
<td>The objective of design-science research is to develop technology-based solutions to important and relevant business problems.</td>
</tr>
<tr>
<td>Guideline 3: Design Evaluation</td>
<td>The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.</td>
</tr>
<tr>
<td>Guideline 4: Research Contributions</td>
<td>Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies.</td>
</tr>
<tr>
<td>Guideline 5: Research Rigor</td>
<td>Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.</td>
</tr>
<tr>
<td>Guideline 6: Design as a Search Process</td>
<td>The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.</td>
</tr>
<tr>
<td>Guideline 7: Communication of Research</td>
<td>Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.</td>
</tr>
</tbody>
</table>
### Table 3. Design Science Artifacts (quoted from March & Smith (1995))

<table>
<thead>
<tr>
<th>ARTIFACT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Construct</td>
<td>Construct or concepts form the vocabulary of a domain. They constitute a conceptualization used to describe problems within the domain and to specify their solutions. (p. 256)</td>
</tr>
<tr>
<td>Model</td>
<td>A model is a set of propositions or statements expressing relationships among constructs. In design activities, models represent situations as problem and solution statements. (p. 256)</td>
</tr>
<tr>
<td>Method</td>
<td>A method is a set of steps (an algorithm or guideline) used to perform a task. (p. 257)</td>
</tr>
<tr>
<td>Instantiation</td>
<td>An instantiation is the realization of an artifact in its environment. (p. 258)</td>
</tr>
</tbody>
</table>

Peffers et al. (2008) addressed design science and proposed a design science research methodology for Information Systems research comprised of six activities. In the paper, the authors argued that the Information Systems discipline was lacking a method for design science research and therefore proposed one. In Table 4, each activity is shortly addressed, quoting the work from Peffers et al. (2008).

### Table 4. Design Science Research Methodology (quoted from Peffers et al. (2008))

<table>
<thead>
<tr>
<th>ACTIVITY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Problem identification and motivation.</td>
<td>Define the specific research problem and justify the value of solution. (p. 52)</td>
</tr>
<tr>
<td>2: Define the objectives for a solution</td>
<td>Infer the objectives of a solution from a problem definition and knowledge of what is possible and feasible. (p. 55)</td>
</tr>
<tr>
<td>3: Design and development</td>
<td>Create the artifact. Such artifacts are potentially constructs, models, methods, or instantiations […] (p. 55)</td>
</tr>
<tr>
<td>4: Demonstration</td>
<td>Demonstrate the use of the artifact to solve one or more instances of the problem. (p. 55)</td>
</tr>
<tr>
<td>5: Evaluation</td>
<td>Observe and measure how well the artifact supports a solution to the problem. (p. 56)</td>
</tr>
<tr>
<td>6: Communication</td>
<td>Communicate the problem and its importance, the artifact, its utility and novelty, the rigor of its design, and its effectiveness to researchers and other relevant audiences such as practicing professionals, when appropriate. (p. 56)</td>
</tr>
</tbody>
</table>

In the paper, the authors also pointed out that their methodology is flexible and iterative. An advantage to this method is that the researcher, depending on focus, can start in different activities. In other words, a problem-centered
initiation starts in activity one, an objective-centered solution starts in activity two, a design & development centered initiation starts in activity three and finally a client/context initiated research entry point in activity four.

As can be seen by the descriptions of each activity, the majority correspond to one or two of the guidelines proposed by Hevner et al. (2004) which indicates how they fit together. The method used in this thesis could be described and viewed as a combination of the guidelines proposed in Havner et al. (2004) and the method proposed in Peffers et al. (2008). The research has, in general, followed the sequence in Peffers et al. (2008), although not in a strict sequential way but rather in a repetitive way focusing on different issues and phases. In addition the guidelines proposed in Hevner et al. (2004) have also been guiding the research. More precisely, this means that artifacts have been designed (guideline one), the designed artifacts have been evaluated (guideline three) and so on. Other slightly different approaches to the design science paradigm have been proposed and developed (see for instance Carlsson (2006), Henningsson (2008) and Hrastinski (2007)). In the three publications, the authors address the need to extend the design science paradigm to include people and organizations referring to Herbert Simon’s book (1969) where the author stated that:

Everyone designs who devises courses of action aimed at changing existing situations into preferred ones. (p. 55)

Yet, in this thesis, I have focused on the artifact per se following the work proposed by March & Smith (1995) later refined by Hevner et al. (2004). Finally, before moving on to the analysis of the papers upon which this thesis is based and their contributions, two things need to be addressed in connection to design science research: research contributions and research evaluation methods. Regarding research contributions, Hevner et al. (2004) stated (see guideline four) that:

Design-science research holds the potential for three types of research contributions based on the novelty, generality, and significance of the designed artifact. […]

1. The Design Artifact. Most often, the contribution of design-science research is the artifact itself. […]
2. Foundations. The creative development of novel, appropriately evaluated constructs, models, methods, or instantiations that extend and improve the
existing foundations in the design-science knowledge base are also important contributions.

3. Methodologies. Finally, the creative development and use of evaluation methods (e.g., experimental, analytical, observational, testing, and descriptive) and new evaluation metrics provide design-science research contributions. (p. 87)

As will be shown in the following chapters, the research contributions and results of the work reported in this thesis are mainly placed within the second type of contributions: Foundations. Regarding evaluation methods, Hevner et al. (2004) stated (see guideline three) that:

The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods. Evaluation is a crucial component of the research process. (p. 85)

Hevner et al. (2004) also argue that at least five types of design evaluation methods are usable: observational, analytical, experimental, testing and descriptive (see Table 5). As will be shown in the following chapters, the evaluation methods used in this thesis are analytical and descriptive.
Table 5. Design Evaluation Methods (quoted from Hevner et al. (2004 p. 86))

<table>
<thead>
<tr>
<th>EVALUATION METHOD</th>
<th>DESCRIPTION</th>
</tr>
</thead>
</table>
| 1. Observational  | Case Study: Study artifact in depth in business environment  
Field Study: Monitor use of artifact in multiple projects |
| 2. Analytical     | Static Analysis: Examine structure of artifact for static qualities (e.g., complexity)  
Architecture Analysis: Study fit of artifact into technical IS architecture  
Optimization: Demonstrate inherent optimal properties of artifact or provide optimality bounds on artifact behavior  
Dynamic Analysis: Study artifact in use for dynamic qualities (e.g., performance) |
| 3. Experimental   | Controlled Experiment: Study artifact in controlled environment for qualities (e.g., usability)  
Simulation: Execute artifact with artificial data |
| 4. Testing        | Functional (Black Box) Testing: Execute artifact interfaces to discover failures and identify defects  
Structural (White Box) Testing: Perform coverage testing of some metric (e.g., execution paths) in the artifact implementation |
| 5. Descriptive    | Informed Argument: Use information from the knowledge base (e.g., relevant research) to build a convincing argument for the artifact's utility  
Scenarios: Construct detailed scenarios around the artifact to demonstrate its utility |

In the tables that follow each paper and research study is summarized. The purpose is to provide an overview of each paper with focus on research method and research contributions. It should be noted that the research studies are presented in order of acceptance. The order of acceptance is not necessarily the order of publication since conferences print and publishes the proceedings up to two years after the conference has taken place.
Table 6. Overall presentation of research study one (Bellström & Carlsson 2004)

<table>
<thead>
<tr>
<th>TITLE</th>
<th>TOWARDS AN UNDERSTANDING OF THE MEANING AND THE CONTENTS OF A DATABASE THROUGH DESIGN AND RECONSTRUCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Research question</td>
<td>What happens and what are the consequences when one concept name (label) is chosen for several concept names (labels) while resolving synonyms (similarities) in schema integration?</td>
</tr>
<tr>
<td>Research approach and method</td>
<td>The purpose of this paper was to analyze and describe what happens when one concept name (label) is chosen to represent two concept names (labels). Choosing one concept name (label) results in compression (merge) of two concept names (labels), a problem named concept name compression. In the paper, we study the problem from a new perspective focusing on maintaining the vocabulary used in the source schemata.</td>
</tr>
<tr>
<td>Premises for the research</td>
<td>In schema integration, synonyms (similarities) between two source schemata have to be recognized and resolved before merging the schemata into one global schema. However, if designers and users make wrong decisions or even try to simplify the source schemata or integrated global schema too much they might instead lose one or several concept names (labels). In the paper we analyze, discuss and describe problems and pitfalls that might be the result of such incorrect decisions.</td>
</tr>
<tr>
<td>Contributions and results</td>
<td>The main contribution of this paper is given in the analysis of concept name compression. Concept name compression occurs if several concept names (labels) are compressed (merged) into one concept name (label), for example when choosing one concept name (label) to resolve synonyms (similarities) that have been recognized between two source schemata. Concept name compression means that one concept name (label) is lost. In other words, two concept names (labels) are compressed (merged) into one concept name (label). The two proposed generic methods that are applicable if concept name compression has occurred are also contributions of the paper. However, it should be noted that the main contribution of the paper is still given in the analysis of concept name compression. Placing the contributions of this paper in connection to design science research, the main contribution is classified as a construct while the two proposed methods are classified as methods.</td>
</tr>
<tr>
<td>Implications</td>
<td>If users should be able to verify or decline the integrated global conceptual schema, they should, in the integrated schema, see their own concept names – their vocabulary. However, if one concept name is lost (compressed with another), the user might not be able to verify or decline the schema. If one concept name in the source schemata is lost, the vocabulary in the integrated schema is also impoverished.</td>
</tr>
<tr>
<td>Research topic</td>
<td>Maintaining Vocabulary in Schema Integration (RT1)</td>
</tr>
<tr>
<td>Authors and responsibility</td>
<td>Peter Bellström (70 %) and Sten Carlsson (30 %)</td>
</tr>
<tr>
<td>Table 7. Overall presentation of research study two (Bellström 2005)</td>
<td></td>
</tr>
<tr>
<td>---------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>Title</strong></td>
<td>USING ENTERPRISE MODELING FOR IDENTIFICATION AND RESOLUTION OF HOMONYM CONFLICTS IN VIEW INTEGRATION</td>
</tr>
<tr>
<td><strong>Research question</strong></td>
<td>How can homonyms between two source schemata be recognized and resolved in schema integration?</td>
</tr>
<tr>
<td><strong>Research approach and method</strong></td>
<td>The purpose with this paper was twofold. First, to propose and describe a method for recognizing homonyms in schema integration and second, to propose, describe and illustrate a method for resolving homonyms recognized in schema integration while applying the notation given in the Enterprise Modeling approach. A comparative literature review on homonym resolution methods was conducted. An overview on how to resolve similarities and differences between two source schemata applying the notation given in the Enterprise Modeling approach was also provided.</td>
</tr>
<tr>
<td><strong>Premises for the research</strong></td>
<td>In schema integration, it is important to both recognize and resolve homonyms between two source schemata before merging them into one schema. However, methods for recognition of homonyms often only focus on comparing the concept names (labels) without comparing the neighborhood of the concepts. Otherwise most methods for the resolution of homonyms use renaming, which might cause problems such as semantic loss. In this paper, both a method for recognition and a method for resolution of homonyms are proposed and demonstrated. Both methods are adapted to fit the notation of the Enterprise Modeling approach.</td>
</tr>
<tr>
<td><strong>Contributions and results</strong></td>
<td>This paper has two main contributions. The first is given in the method for recognizing homonyms in schema integration and the second in the method for resolving homonyms in schema integration. Placing the contributions of this paper one in connection to design science research, both are classified as methods.</td>
</tr>
<tr>
<td><strong>Implications</strong></td>
<td>By applying the proposed recognition method, it is possible to identify homonyms that otherwise might pass unnoticed. The motivation for this is that both concept names and concept neighborhoods are compared and analyzed. By applying the proposed resolution method, concept names (labels) are retained and the schema is semantically enriched. The motivation for this is that both a stronger dependency and the context of the concept are included in the resolution method.</td>
</tr>
<tr>
<td><strong>Research topic</strong></td>
<td>Integration of Static Schemata (RT2)</td>
</tr>
<tr>
<td><strong>Authors and responsibility</strong></td>
<td>Peter Bellström (100%)</td>
</tr>
</tbody>
</table>
### Table 8. Overall presentation of research study three (Bellström & Jakobsson 2006)

| TITLE | TOWARDS A GENERIC AND INTEGRATED ENTERPRISE MODELING APPROACH TO DESIGNING DATABASES AND SOFTWARE COMPONENTS |
|-------|------------------------------------------------------------------------------------------------|---|
| Research question | How can a generic method for integration of static and dynamic schemata be designed and defined? |
| Research approach and method | The purpose with this paper was to analyze, describe and illustrate a generic method for designing and integrating static and dynamic schemata while applying the notation given in the Enterprise Modeling approach. The problem was studied using a set of schemata representing a sales and a warehouse department. The example schemata were used during both the schema design phase and the schema integration phase thereby illustrating and describing how the proposed generic method could empower not only the schema design phase but also the schema integration phase. |
| Premises for the research | Both schema design and schema integration are important and critical phases when developing databases and information systems. However, how to integrate the static schemata, the dynamic schemata are the static and dynamic schemata together are problems seldom addressed. In the paper, all of these issues were addressed while applying the notation in the Enterprise Modeling approach. |
| Contributions and results | The main contribution of this paper is given in the generic method to designing and integrating static and dynamic schemata. In the method, the static schemata are first designed followed by the dynamic schemata. When the schemata have been designed they are also integrated. Integration is conducted by first integrating the static schemata resulting in one global conceptual database schema. This is followed by integrating the dynamic schemata resulting in one global schema comprised of both the dynamic and the static aspects. Placing the contribution of this paper in connection to design science research, the contribution is classified as a method. |
| Implications | By applying the generic method for designing and integrating static and dynamic schemata, it is possible to validate the static aspects while designing and integrating the dynamic aspects. This is possible since the designer and user can first design the static schemata followed by the dynamic schemata. After that, the designer and user can integrate the static schemata followed by integrating the dynamic schemata. This should, in the end, result in one global integrated schema containing both the static and the dynamic aspects of the database. |
| Research topic | Integration of Static and Dynamic Schemata (RT3) |
| Authors and responsibility | Peter Bellström (55 %) and Lars Jakobsson (45 %) |
Table 9. Overview presentation of research study four (Bellström 2006b)

<table>
<thead>
<tr>
<th>TITLE</th>
<th>BRIDGING THE GAP BETWEEN COMPARISON AND CONFORMING THE VIEWS IN VIEW INTEGRATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Research question(s)</td>
<td>Why, how and when should inference rules be applied in schema integration?</td>
</tr>
<tr>
<td>Research approach and method</td>
<td>The purpose with this paper was threefold. First, to analyze and describe why inference rules should be applied in schema integration; second, to analyze and describe how inference rules should be applied in schema integration and thirdly, to analyze when inference rules should be applied in schema integration. The how question was the most important one, since addressing its issues in relation to the two most important phases of schema integration, comparison of the schemata and conforming the schemata were also addressed.</td>
</tr>
<tr>
<td>Premises for the research</td>
<td>Schema integration is a complex, time-consuming and error-prone task. Methods that facilitate schema integration are therefore needed. One such method is inference rules. In this paper, inference rules are analyzed, discussed and described in relation to the why, how and when questions. While analyzing and answering the how question, the notation given in the Enterprise Modeling approach was applied.</td>
</tr>
</tbody>
</table>
| Contributions and results | The paper has two main contributions. The first is given in the results of the analysis of the why question: inference rules should be applied to facilitate schema integration by providing a set of reasoning rules. The second contribution is given in the results of the analysis of the how question: to deduce new concepts and/or dependencies where similarities or differences have been recognized between two source schemata. Besides these two contributions, the discussion regarding when to apply inference rules could also be viewed as a contribution. However, this is a much smaller contribution compared to the other two and therefore not discussed further in this overall presentation. 

Placing the contributions of this paper in connection to design science research, the first contribution is classified as a construct and the second as a method. |
| Implications | By applying inference rules, reasoning rules, in schema integration, both recognition and resolution of similarities and differences between two source schemata are facilitated. This is the case since an inference rule is comprised of an if (recognition) and a then (resolution) part. By applying inference rules it is also possible to bridge the two most important, difficult and critical phases in schema integration. |
| Research topic | Integration of Static Schemata (RT2) |
| Authors and responsibility | Peter Bellström (100%) |
Table 10. Overall presentation research study five (Bellström & Carlsson 2006)

<table>
<thead>
<tr>
<th>TITLE</th>
<th>LANGUAGE ASPECTS OF CONCEPTUAL DATABASE DESIGN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Research question</td>
<td>How can synonyms (similarities), hypernyms-hyponyms (similarities) and homonyms (differences) be resolved during schema integration without causing concept name compression?</td>
</tr>
<tr>
<td>Research approach and method</td>
<td>The purpose with this paper was to propose, illustrate and analyze resolution methods for synonyms, hypernyms-hyponyms and homonyms. The proposed resolution methods should not cause concept name compression since concept names should be retained even after resolving synonyms, hypernyms-hyponyms and homonyms. The problem is studied from a new perspective focusing on how to resolve similarities and differences between two source schemata without causing concept name compression and without impoverishing the vocabulary used in the source schemata and/or integrated schema.</td>
</tr>
<tr>
<td>Premises for the research</td>
<td>In schema integration, both similarities, such as synonyms and hypernyms-hyponyms, and differences, such as homonyms, between two source schemata have to be resolved before merging the schemata into one global schema. However, traditional resolution methods focus on renaming one or two concept names. Renaming might cause concept name compression. In this paper, new resolution methods are proposed and demonstrated, by applying the notation given in the Enterprise Modeling approach.</td>
</tr>
<tr>
<td>Contributions and results</td>
<td>The main contribution of this paper is given in three resolution methods for synonyms, hypernyms-hyponyms and homonyms that do not cause semantic loss. The analysis of how to maximize the line of mutual interpretation is also a contribution of the paper. Nevertheless, the main contribution is still given in the three resolution methods. Placing the contributions of this paper in connection to design science research, the main contribution is classified as method while the analysis of how to maximize the line of mutual interpretation is classified as construct.</td>
</tr>
<tr>
<td>Implications</td>
<td>By applying the proposed resolution methods for synonyms, hypernyms-hyponyms and homonyms, all concept names (labels) are retained after resolution. This means that the vocabulary of the local schemata is retained even after resolving and merging the schemata. This also means that the users should be able to verify or decline the integrated schema since their vocabulary is retained after resolution.</td>
</tr>
<tr>
<td>Research topics</td>
<td>Maintaining Vocabulary in Schema Integration (RT1) and Integration of Static Schemata (RT2)</td>
</tr>
<tr>
<td>Authors and responsibility</td>
<td>Peter Bellström (70%) and Sten Carlsson (30%)</td>
</tr>
<tr>
<td><strong>Table 11.</strong> Overall presentation of research study six (Bellström et al. 2007)</td>
<td></td>
</tr>
<tr>
<td>---------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>TITLE</strong></td>
<td>RECOGNITION AND RESOLUTION OF LINGUISTIC CONFLICTS: THE CORE TO A SUCCESSFUL VIEW AND SCHEMA INTEGRATION</td>
</tr>
<tr>
<td><strong>Research question</strong></td>
<td>How can similarities (synonyms, hypernyms-hyponyms, holonyms-meronyms) and differences (homonyms) between two source schemata be recognized and resolved in schema integration?</td>
</tr>
<tr>
<td><strong>Research approach and method</strong></td>
<td>The purpose of this paper is twofold. First, to describe and propose a manual method for recognizing similarities (synonyms, hypernyms-hyponyms, holonyms-meronyms) and differences (homonyms) between two source schemata in schema integration. Second, to give an overview and propose manual resolution methods for similarities and differences recognized between two source schemata. A sales department schema and a warehouse department schema served as a small illustrating example for the addressed problems. The notation given in the Enterprise Modeling approach was applied to graphically illustrate the problems of recognizing and resolving similarities and differences between two source schemata.</td>
</tr>
<tr>
<td><strong>Premises for the research</strong></td>
<td>In schema integration, both similarities and differences, in the paper referred to as linguistic conflicts, have to be recognized and resolved before the two source schemata can be merged. Yet, both recognition and resolution of similarities and differences are difficult tasks to perform. In the paper, manual methods to solve the described problems are addressed while applying the notation given in the Enterprise Modeling approach.</td>
</tr>
<tr>
<td><strong>Contributions and results</strong></td>
<td>The main contributions of this paper are given in the methods for recognizing and resolving similarities and differences (in the paper referred to as linguistic conflicts) between two source schemata.</td>
</tr>
<tr>
<td><strong>Implications</strong></td>
<td>While conducting schema integration on an implementation independent level, relevant methods for recognition and resolution of similarities and differences between two source schemata should be used. However, most of the methods proposed in the literature deal instead with implementation dependent schema integration. Applying the notation given in the Enterprise Modeling approach the proposed recognition and resolution methods should facilitate and improve schema integration. This is the case since the methods should firstly be applied on a modeling approach focusing on an implementation independent level, meaning focusing only on concepts (including their names) and dependencies (connections/links) between them.</td>
</tr>
<tr>
<td><strong>Research topic</strong></td>
<td>Integration of Static Schemata (RT1)</td>
</tr>
<tr>
<td><strong>Authors and responsibility</strong></td>
<td>Peter Bellström (40 %), Jürgen Vöhringer (40 %) and Alexander Salbrechter (20 %)</td>
</tr>
</tbody>
</table>
Table 12. Overall presentation of research study seven (Jakobsson & Bellström 2007)

<table>
<thead>
<tr>
<th>TITLE</th>
<th>DESIGNING SOFTWARE COMPONENTS FOR DATABASE CONSISTENCY – AN ENTERPRISE MODELING APPROACH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Research question</td>
<td>How can legal states and state changes of the database be illustrated and integrated to ensure database consistency?</td>
</tr>
<tr>
<td>Research approach and method</td>
<td>The purpose with this paper was to analyze and describe how legal states and state changes of the database could be represented, illustrated and integrated in the conceptual database schema. The problem was studied using a set of schemata representing a sales and warehouse department. In the example schemata, we focused on states in connection with successful and unsuccessful database actions (transactions). To represent and illustrate the successful and unsuccessful database actions, states and state changes, we applied the notation given in the Enterprise Modeling approach.</td>
</tr>
<tr>
<td>Premises for the research</td>
<td>Databases are serving enterprises with data that are interpreted to information and used on a daily basis. It is therefore very important that the database stores the right information and that the database is in a consistent state. This is also an important issue to address while designing (modeling) the database. However, states and state changes should be addressed while conducting conceptual database design and not on an ad hoc manner while implementing the database. Consequently, avoiding inconsistency issues is very important already on the conceptual level independent of implementation. In this paper, we applied the notation given in the Enterprise Modeling approach to show how legal states and state changes could be represented, illustrated and integrated in the conceptual database schema.</td>
</tr>
<tr>
<td>Contributions and results</td>
<td>The main contribution of this paper is given in the method on how to design and illustrate legal states and state changes of the database. By applying the notation in the Enterprise Modeling approach, we are able to represent legal states and state changes at pre- and post conditions in the conceptual database schema. This also means that to ensure a consistent database it is possible to, on a conceptual level, use the same modeling constituents, states (conditions), while designing and integrating both static and dynamic schemata. Applying the same constituents also facilitates schema integration and the ability to reach a consistent global integrated conceptual database schema.</td>
</tr>
<tr>
<td>Implications</td>
<td>By applying the proposed method on how to design and represent legal states and state changes of the database, we are able to analyze and illustrate legal states and state changes in a systematic way representing these as pre- and post-conditions. In doing so, we are also able to address the problem of inconsistency very early in the development process. This also means that while integrating the static and dynamic schemata, we use pre- and post-conditions for both legal states and state changes of the database and legal states and state changes of the application accessing the database.</td>
</tr>
<tr>
<td>Research topic</td>
<td>Integration of Static and Dynamic Schemata (RT3)</td>
</tr>
<tr>
<td>Authors and responsibility</td>
<td>Peter Bellström (45 %) and Lars Jakobsson (55 %)</td>
</tr>
</tbody>
</table>
Table 13. Overall presentation of research study eight (Bellström 2009)

<table>
<thead>
<tr>
<th>Title</th>
<th>ON THE PROBLEM OF SEMANTIC LOSS IN VIEW INTEGRATION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Research questions</strong></td>
<td>What differentiates and relates information loss, concept name compression and semantic loss? Does the strive for minimality, the use of different modeling constructs or the use of non-comparable transformations cause semantic loss?</td>
</tr>
<tr>
<td><strong>Research approach and method</strong></td>
<td>The purpose with this paper was to analyze how information loss, concept name compression and semantic loss relate and/or differentiate to each other. A second purpose was to analyze if minimality, the use of different modeling constructs and the use of non-comparable transformations might cause semantic loss. A literature review over resolution methods for the mentioned problems and issues was conducted. A set of conceptual database schemata for representing “Publications” also served as an illustrating example for problems that have to be dealt with in schema integration.</td>
</tr>
<tr>
<td><strong>Premises for the research</strong></td>
<td>In schema integration, both similarities and differences have to be recognized and resolved before the source schemata can be merged into one global schema. However, while resolving similarities and differences bad resolution methods might instead cause several problems such as information loss, concept name compression and semantic loss. Striving towards a minimal schema, using different modeling constructs or using non-comparable transformations are also issues that are important to consider in schema integration. In this paper, the three problems are analyzed aiming to identify what differentiates and relates these terms. As well it is also analyzed if minimality, the use of different modeling constructs and the use of non-comparable transformations might cause semantic loss.</td>
</tr>
<tr>
<td><strong>Contributions and results</strong></td>
<td>The main contribution of this paper is given in the results of the analysis of information loss, concept name compression and semantic loss and how these differentiate and relate. The analysis as such should also be viewed as a part of the contribution. The analysis on minimality, the use of different modeling constructs and the use of non-comparable transformations and if these might cause semantic loss is also a contribution of the paper. Placing the contributions of this paper in connection to design science research, the main contribution is classified as a construct and the result of the second analysis as a construct.</td>
</tr>
<tr>
<td><strong>Implications</strong></td>
<td>While conducting schema integration on an implementation independent level, it is important to choose resolution methods that do not cause problems in relation to the vocabulary expressed in the source schemata. In this paper, three such problems are analyzed according to differences and similarities. The conclusion is that all of them should be avoided since they might cause problems in relation to the used vocabulary. In the paper three issues that are often addressed in schema integration methods and approaches where also analyzed in relation to semantic loss. It was concluded that minimality, the use of different modeling constructs and the use of non-comparable transformations all cause semantic loss and should therefore be avoided.</td>
</tr>
<tr>
<td><strong>Research topics</strong></td>
<td>Maintaining Vocabulary in Schema Integration (RT1) and Integration of Static Schemata (RT2)</td>
</tr>
<tr>
<td><strong>Authors and responsibility</strong></td>
<td>Peter Bellström (100 %)</td>
</tr>
</tbody>
</table>
Research question: Which approaches have been proved to be useful and successful while conducting integration of dynamic schemata?

Research approach and method: The purpose with this paper was to study which integration approaches have been proved to be not only useful but also successful while conducting integration of dynamic schemata. The problem was studied by analyzing existing literature on the topic of integration of dynamic schemata. To be classified as useful and successful, the integration approach should be generic, meaning it should be modeling language independent focusing on conditions (states) and actions.

Premises for the research: Integration of dynamic schemata is a very complex and time consuming task that poses major problems for the modeling community. The absence of a common strategy while modeling dynamic schemata is emphasized when comparing integrated static schema and integrated dynamic schema. Most modeling languages for static modeling follow the strategy that objects can be classified to classes or entity types. These are connected to each other by relationships and each object might also have properties. On the other hand, modeling languages for dynamic schemata differ greatly and in the paper we therefore argued that conditions (states) and actions are the minimal set of constituents used to model the dynamic aspects.

Contributions and results: The main contribution of this paper is given in the six generic guidelines for integration of dynamic schemata: (1) performing schema integration on the pre-design level, (2) computer supported integration utilizing user feedback, (3) using domain repositories for supporting the integration process, (4) standardizing concept notions and utilizing them during integration, (5) neighborhood-based conflict recognition and (6) pattern-based resolution of integration conflicts.

Implications: Integration of dynamic schema is a complex task and therefore methods, approaches and guidelines that support integration are needed. In this paper, we propose and describe six generic guidelines that facilitate integration of dynamic schemata. However, it should be noted that several of the proposed and described guidelines are also applicable while conducting integration of static schemata.

Research topic: Integration of Static and Dynamic Schemata (RT3)

Authors and responsibility: Peter Bellström (40%), Jürgen Vöhringer (40%) and Christian Kop (20%)
### Table 15. Overall presentation of research study ten (Bellström & Jakobsson 2008)

<table>
<thead>
<tr>
<th>TITLE</th>
<th>ANALYZING AND DEFINING ACTIVE DATABASE RULES ON A CONCEPTUAL LEVEL.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Research question</td>
<td>How can active database rules be illustrated and integrated to ensure database consistency?</td>
</tr>
<tr>
<td>Research approach and method</td>
<td>The purpose with this paper was to analyze and describe how active database rules could be illustrated and integrated in the conceptual database schema. An active database rule is, in this paper, expressed as an Event-Condition-Action rule (ECA-rule), meaning if a specific event (E) occurs and the defined condition (C) evaluates to true then the action (A) is executed. For this paper, we developed a MySQL database that included tables, a stored procedure and a trigger (ECA-rule). We also developed a web-based PHP-application and a standalone Java-application both accessing the MySQL database and triggering the trigger. Having done that, we conducted reverse engineering, focusing on the MySQL database while applying the notation given in the Enterprise Modeling approach.</td>
</tr>
<tr>
<td>Premises for the research</td>
<td>Traditionally, conceptual database design has focused on defining the static aspects of the future database. This is in agreement with how databases have been viewed – as passive data repositories. However, nowadays, database management systems are able to automatically respond to events; they are more active. Active database rules should be described and illustrated early in the development process and not addressed in an ad hoc manner while implementing the database. Methods and modeling approaches that deal with this are therefore needed. In this paper, we used the notation given in the Enterprise Modeling approach to show how active database rules could be represented and integrated in the conceptual database schema.</td>
</tr>
<tr>
<td>Contributions and results</td>
<td>The main contribution of this paper is given in the method on how to design and illustrate active database rules (ECA-rules) in conceptual database design. By applying the notation in the Enterprise Modeling approach we were able to represent active database rules as actions and pre- and post-conditions (states) in the conceptual database schema. This also means that to ensure a consistent database it is possible, on a conceptual level, to use the same modeling constituents, pre- and post-conditions together with actions, while designing and integrating static and dynamic aspects of the database. Applying the same constituents also facilitates schema integration and the ability to reach a consistent global integrated conceptual database schema. Placing the contribution of this paper in connection to design science research, the contribution is classified as a method.</td>
</tr>
<tr>
<td>Implications</td>
<td>By applying the proposed method on how to design and represent active database rules, we are able to analyze and illustrate ECA-rules in a systematic way representing these as actions and pre- and post-conditions. In doing so, we are also able to address the problem of inconsistency very early in the development process, meaning already while doing conceptual database design. This also means that while integrating the static and dynamic schemata we use actions and pre- and post-conditions for both ECA-rules in the database and pre- and post-conditions for legal states and state changes of the application accessing the database.</td>
</tr>
<tr>
<td>Research topic</td>
<td>Integration of Static and Dynamic Schemata (RT3)</td>
</tr>
<tr>
<td>Authors and responsibility</td>
<td>Peter Bellström (60 %) and Lars Jakobsson (40 %)</td>
</tr>
</tbody>
</table>
**Table 16.** Overall presentation of research study eleven (Bellström et al. 2009)

<table>
<thead>
<tr>
<th>TITLE</th>
<th>TOWARD MODELING-LANGUAGE INDEPENDENT INTEGRATION OF DYNAMIC SCHEMATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Research question</td>
<td>What phases are needed for the integration of dynamic schemata made up of actions and conditions? In what phase(s) should each of the six generic guidelines proposed and described in Bellström et al. (2008) be applied?</td>
</tr>
<tr>
<td>Research approach and method</td>
<td>The purpose with this paper was to study what phases are needed for the integration of dynamic schemata made up of actions and pre- and post-conditions. A second purpose was to study the results of the first question and relate the proposed phases to the generic guidelines proposed in Bellström et al. (2008). The problem was studied using a set of schemata representing two different book reservations in a library system. Three modeling languages, the Enterprise Modeling approach, Klagenfurt Conceptual PreDesign Model and Adonis were used to show and illustrate how the proposed phases (the method) could be applied.</td>
</tr>
<tr>
<td>Premises for the research</td>
<td>Integration of dynamic schemata is a complex task. Therefore, methods and approaches to facilitate integration are needed. However, one issue that has to be addressed while conducting integration of dynamic schemata is the difference between different modeling languages. In this paper, we therefore assume that each modeling language developed to address the dynamic aspects could be reduced to actions and conditions. While studying what phases a method for integration of dynamic schemata should be comprised of, the generic guidelines proposed in Bellström et al. (2008) were applied. Finally, three modeling languages are applied to show that actions and conditions are the minimal set of constituents used for dynamic modeling. These modeling languages are the Enterprise Modeling Approach, Klagenfurt Conceptual PreDesign Model and Adonis.</td>
</tr>
<tr>
<td>Contributions and results</td>
<td>The main contribution of this paper is given in the method, set of phases, applicable for the integration of dynamic schemata comprised of actions and states. The contribution also includes the mapping of the generic guidelines proposed in Bellström et al. (2008) to the proposed phases of the integration method. Placing the contribution of this paper in connection to design science research, the contribution is classified as a method.</td>
</tr>
<tr>
<td>Implications</td>
<td>By applying the proposed method, set of phases, we are able, in a systematic way, to integrate dynamic schemata comprised of actions and conditions (states). Focussing on actions and conditions, the proposed method could be viewed as modeling language independent. This is supported by the fact that actions and conditions are identified as the minimal subset of constituents used in any dynamic modeling language.</td>
</tr>
<tr>
<td>Research topic</td>
<td>Integration of Static and Dynamic Schemata (RT3)</td>
</tr>
<tr>
<td>Authors and responsibility</td>
<td>Peter Bellström (40%), Jürgen Vöhringer (40%) and Christian Kop (20%)</td>
</tr>
</tbody>
</table>
2.6 Research Process

Besides giving a clear description of the chosen research method, such as in this thesis design science, it is also important to give a clear description of the research process. Here it is important to point out that the research process has to be clearly “marked out” (e.g. Frenckner 1986) because the reader has to be able to follow the research process and the progression of the presented work. In other words, the research process has to be traceable. To achieve traceability, one has to clearly indicate, for each of the included papers, the order of acceptance and which of the research topics of the thesis the paper belongs. Both these issues are addressed in the tables presented in the former section.

Besides traceability, it is important to clearly state links and overlaps between the presented research topics. In Figure 2, the three research topics studied in this thesis are represented as circles in a Venn diagram. Between each research topic a link and overlap exists. In Figure 2, this is illustrated as the “star” in the intersection between the research topics. In the middle of Figure 2 (dotted area) the intersection between all three research topics is illustrated, meaning all three research topics have one very important issue in common: schema integration.

![Figure 2. Links and overlaps between research topics](image)

The intersection between research topic one, *Maintaining Vocabulary in Schema integration*, and research topic two, *Integration of Static Schemata*, should be interpreted as follows: while resolving similarities and differences between two
static source schemata, it is important that the chosen resolution methods do not cause semantic loss.

The intersection between research topic two, Integration of Static Schemata, and research topic three, Integration of Static and Dynamic Schemata, should be interpreted as follows: while integrating two static and dynamic source schemata, it is important that the static source schemata have already been merged and similarities and differences between the source schemata have been resolved. This is important because when integrated, the static and dynamic source schemata are very closely connected, not only by the use of concept names (object) [state], but also because the structure of the static source schema is included in the dynamic source schema.

The intersection between research topic three, Integration of Static and Dynamic Schemata, and research topic one, Maintaining Vocabulary in Schema integration, should be interpreted as follows: while integrating static and dynamic source schemata it is important that the chosen resolution methods do not cause semantic loss.

The reasoning behind writing a PhD thesis compiled of already presented and published work (papers) and a cover and summary is that I believe in small task completion. I therefore agree with Davis and Parker (1997) claiming that:

> Individuals tend to be motivated by task completion. In other words, completing a task, however small, provides positive feedback and motivation. (p. 30)

By writing this type of thesis, the quality of the work is also checked since each paper has already been read and reviewed by other researchers in the research field. Therefore, the papers and research have already been exposed to criticism. At the same time, the author and researcher is given feedback that the research and paper has reached a certain level and is therefore acceptable.

Collaborating with other researchers and writing papers together is not only fruitful and important but also necessary to succeed in the research community. However, it should be noted that writing papers together also influences the process that the researcher has to follow while both writing the thesis and while developing skills to become researcher.
During my PhD studies, I have had the opportunity to have three such collaborations, which is reflected in the published papers and the presented research topics. In the first research topic, *Maintaining Vocabulary in Schema Integration*, I collaborated with Sten Carlsson. In the second research topic, *Integration of Static Schemata*, I collaborated with Jürgen Vöhringer and in the third research topic, *Integration of Static and Dynamic Schemata*, I collaborated with Lars Jakobsson and Jürgen Vöhringer. For all three collaborations I am very grateful.

### 2.7 Target Group

The target group of this thesis is primarily researchers within both the database research area and the information systems research area focusing on conceptual modeling and integration. This is due to the fact that this thesis includes both static and dynamic aspects of database design. On the other hand, this work should also be beneficial to practitioners within the database and the information systems area since several concrete proposals to resolve similarities and differences between two source schemata are discussed and illustrated.
3 Modeling Language and Integration Process

The purpose of this chapter is to present an overview of the four main modeling languages referenced in this thesis and to present an overview of the schema integration process. The chapter is therefore divided into two sections. In section one, the Entity-Relationship Modeling Language, the Unified Modeling Language, the Klagenfurt Conceptual Pre-Design Model and the Enterprise Modeling approach are described. In section two, the schema integration process is described and divided into integration of static schemata, integration of dynamic schemata and integration of static and dynamic schemata.

3.1 Modeling Languages

Several modeling languages and approaches have been proposed during the last thirty years. To give an overview, even a short one, of all of them is a difficult and almost impossible task. Consequently, the purpose of this section is to introduce four modeling languages, the Entity-Relationship Modeling Language (Chen 1976), the Unified Modeling Language (Object Management Group 2009), the Klagenfurt Conceptual Pre-Design Model (Fiedl et al. 2000) and the Enterprise Modeling approach (Gustas & Gustiené 2004). These four modeling languages have been chosen since they have either influenced the conceptual database design research field or and partially influenced the content of this thesis one way or another. The Entity-Relationship Modeling Language (ERML), the Unified Modeling Language (UML) and the Klagenfurt Conceptual Pre-Design Model (KCPM) are introduced and described in a brief manner while the Enterprise Modeling approach (EM) is discussed and described on a more detailed level. The four above mentioned modeling languages were chosen since ERML or some extension of it has been viewed as a de facto standard (Spaccapietra & Parent 1994) for conceptual database design. In addition, ERML has dominated schema integration research since the late 1980’s (Song 1995). UML, on the other hand, is nowadays treated as the de facto standard for object oriented modeling (Siau & Lee 2004) and has almost reached the same position for software component design (Cheesman & Daniels 2001) as ERML has for conceptual database design. KCPM has been included because it can be applied to describe the future database and information system on a high level of abstraction (Kop & Mayr 1998), similar to EM, and because it is mentioned and applied in several of the included papers of this thesis (see Bellström et al. (2007, 2008, 2009)).
In this section the Enterprise Modeling approach is discussed and described in more detail compared to the other three modeling languages. The motivation for this is that in this thesis the notation given in EM has been chosen and used while conducting conceptual design of information systems, especially conceptual database design. Finally, while introducing each of the modeling languages, both benefits and shortcomings are addressed for each modeling language and approach.

3.1.1 Entity-Relationship Modeling Language

The Entity-Relationship Modeling Language (ERML) was first proposed by Peter Chen (Chen 1976) in the mid 1970’s and has since become one of the most popular and commonly used modeling languages (e.g. Ambrosio et al. 1995; Batini et al. 1992; Date 2004; Fahrner & Vossen 1995; Spaccapietra & Parent 1994; Storey et al. 1995) for conceptual database design. In an empirical study performed in Australia, (Davies et al. (2006)) ERML was placed as the number-one modeling language for conceptual modeling. More precisely, this meant that ERML was mentioned as the most frequently used modeling technique for database design. In the literature, ERML has during the years also been called a de facto standard for conceptual database design (Spaccapietra & Parent 1994) and ERML or some extension of it has also dominated view integration research since the late 1980’s (Song 1995). It is clear that many designers still prefer ERML when conducting conceptual database design (Davies et al. 2006; Shoval & Shiran 1997).

In general, the Entity-Relationship Modeling Language offers primitives for entities, attributes and relationships. In addition, it is also possible to give the cardinality between entities. Examples of proposed extensions are subset hierarchy and generalization hierarchy (Teorey et al. 1986). An entity was described by Chen (1976) as

\[
\text{[\ldots] a ‘thing’ which can be distinctly identified. (p. 10)}
\]

and a relationship as

\[
\text{[\ldots] an association among entities. (p. 10).}
\]

Date (2004), on the other hand, does not use attribute but instead property and describes it as follows
Nevertheless, it is important to mention that a property could also be associated with a relationship. The descriptions given above indicate that ERML is a modeling language that makes it possible to describe and illustrate the static aspects of a database and information system.

There are several reasons behind the popularity of ERML as pointed out in Bellström (2006a):

- ERML is useful for communicating different definitions of data and relationships with the users (Teorey et al. 1986);
- ERML has been extended, e.g. Higher-order Entity-Relationship Model (Thalheim 2000) and used in several methods (Engels et al. 1992);
- ERML is a good modeling language for defining constructs like the unary and ternary relationships (Shoval & Shiran 1997);
- ERML has concepts naturally occurring in database design (Lee & Ling 2003);
- ERML diagrams are simple and intuitive (Badia 2004).

Although ERML has been mentioned as a de facto standard and used in several methods, it has also been criticized. Engels et al. (1992) for instance criticized the general use of the relationship construct. Johannesson (1993) mentioned that ERML is not adequate as a canonical modeling language in schema integration since it is not clear if a type should be transformed into an entity type or a relationship type. Finally, ERML lacks the important instance-of dependency needed to illustrate classification (Bellström 2005).

3.1.2 Unified Modeling Language

The object oriented approach is today the most widely used approach for requirements engineering (Kaindl 2005). For object oriented modeling, the Unified Modeling Language (Object Management Group 2009) is the most popular modeling language and has also been mentioned as the de facto standard (Siau & Lee 2004). Booch et al. (1999) also point out that UML has emerged as the dominant modeling language for software development, requirements engineering and information systems development. In a study performed by Davies et al. (2006), UML was ranked as one of the top 5 modeling approaches used most frequently. In the same study 63 % of the
respondents did not know or did not use UML for conceptual modeling, in spite of the fact that it is very popular and viewed as a de facto standard. The current version of the UML is 2.2 (Object Management Group 2009) and since version 2.0, UML has 13 types of diagrams divided into static, behavior and interaction diagrams. The Object Management Group describes the Unified Modeling Language as follows:

UML 2.0 defines thirteen types of diagrams, divided into three categories: Six diagram types represent static application structure; three represent general types of behavior; and four represent different aspects of interactions […] (Object Management Group 2010)

Booch et al. (1999) point out in connection with an earlier version of UML that

Even though it is expressive, the UML is not difficult to understand and to use. (p. 13).

In the same book, the authors describe UML as a modeling language for visualizing, specifying, constructing and documenting a software-intensive system.

Even though UML has been mentioned as easy to use and understand, it has been criticized for having 13 diagram types. One example of this is given in Siau and Lee (2004), were the authors point out that the use of ‘use-case diagrams’ and its usefulness in requirements engineering has been questioned. It has even been questioned if use-case diagrams should be part of UML.

Another criticism that has been pointed towards the Unified Modeling Language is the absence of a complete agreement on how to control consistency between the diagrams (Jakobsson & Gustas 2004). This problem is also stressed in a panel paper by Hendersson-Sellers (2005) in which Bran Selic points out that determining the inter-schema consistency

[…] is still an unresolved theoretical problem in the general case […] (p. 11).

Kop and Mayr (2002) emphasize that in non-technical environments, UML should not be used because it does not meet the level of interests and capabilities of users. The authors also state that this could cause incomplete requirements and inferior validation on requirements schemata.
In the panel paper by Henderson-Sellers (2005) UML 2.0 was discussed. In the paper, Steven Cook stated that

[…] UML is essentially a standardized abstract and concrete syntax. Although it claims to have a standard semantics, in practice it does not. (p. 6).

In the same paper, Steve Mellor called UML version 2.0 “a hack” and Joaquin Miller compared UML version 1 with UML version 2 and stated that

The result is a much improved notation, without a foundation. (p. 8).

On the other hand, in the same paper Bran Selic stated that

While it is fair to argue that the semantics of UML might be incomplete, imprecise and, perhaps, inconsistent, and it is fair to complain about the organization of the document itself, it is definitely incorrect and unfair to claim that UML has no semantics. (p. 11).

Even though the panel participants had different opinions regarding the use of the UML version 2.0, all of them were positive concerning the included changes.

Finally, even though UML critics have both negative and positive things to say about the modeling language, it is here to stay. Fowler (2004) states its importance to the modeling community using a simile with the history of the tower of Babel. The author writes

Since its appearance in 1997, it has relegated that particular tower of Babel to history. (p. 1).

3.1.3 Klagenfurt Conceptual Pre-Design Model

The Klagenfurt Conceptual Pre-Design Model (KCPM) was developed at the University of Klagenfurt as a part of the NIBA (Natürlichsprachliche Information Bedarfs Analyse) a German acronym for “Natural language requirements analysis” project (Fliedl et al. 2000). It was developed to counter the trend of complex modeling languages by acting as an interlingua between different users. This means that the result of applying it is an intermediate schema placed between natural language texts and more complex conceptual schemata designed with a more traditional modeling language such as UML. By
applying KCPM, technical design decisions and implementation issues are postponed for the succeeding design phases dealing with these issues (Kop & Mayr 1998).

KCPM offers a set of semantic constituents for modeling both the static and the dynamic aspects of an information system. Static schemata consist of thing-types, connection-types and perspectives (Fliedl et al. 2000). However, additional constraints in connection with ‘n-ary’ connections have also been mentioned as a part of the static modeling notions (Kop & Mayr 1998). In Kop and Mayr (2002) and Fliedl et al. (2003), it was argued that actors, objects and conditions (pre- and post) are part of the basic principles for conceptual modeling. Therefore, by applying the notation of KCPM, dynamic schemata only consist of operation-types, cooperation-types and conditions.

The modeling constituents in the Klagenfurt Conceptual Pre-Design Model can be described as follows: thing-types, a generalization of class (drawn as rectangles), are connected via connection-types (drawn as diamonds). Both thing-types and connection-types are given names which are important since the tools developed in connection to KCPM are based heavily on linguistic instruments (e.g. Fliedl et al. 2001, 2007). Between each binary connection it is also possible to give perspectives for each of the related thing-types. For instance, Writer (thing-type) Write (connection-type) Book (thing-type) can be described including the given perspectives as Writer (thing-type) Writes (perspective) Book (thing-type) and the other way around Book (thing-type) Written by (perspective) Writer (thing-type). An example schema is illustrated in Figure 3.

![Figure 3](image-url)
Operation-types (drawn as ellipses) describe the operations the information system is going to perform and also serve as the link between static and dynamic schemata. Each operation-type can be assigned parameters and actors, which are described as thing-types. On the other hand, cooperation-types (drawn as rectangles around the operation-types) instead group operation-types with the same pre-conditions. Pre- and post-conditions (drawn as small circles with arrows pointing from and to an operation- or cooperation-type) are used to illustrate conditions that are true before/after certain cooperation-types are executed. An example schema is illustrated in Figure 4.

It is worth pointing out that by applying KCPM, it is possible to use both the schema approach and a glossary-like approach. In the glossary-like approach, concepts, together with important metadata, are instead written in columns and rows like a matrix pattern (see Kop & Mayr (1998)). Finally, by applying KCPM, static and dynamic issues are divided into two schema types. However, some consistency checks have been implemented in the NIB.A toolset. The NIB.A toolset is a collection of prototype applications developed for tasks such as generating schemata from requirement texts using the notation in the Klagenfurt Conceptual Pre-Design Model and mapping these to diagrams described using the Unified Modeling Language.
3.1.4 The Enterprise Modeling Approach

The notation of the Enterprise Modeling approach (EM) applied and adopted in this thesis builds upon the work conducted by Professor Remigijus Gustas (e.g., Gustas 1998, 2005; Gustas & Gustienė 2004, 2008).

By applying EM, several aspects of an enterprise are described using the notation of a chosen modeling language developed for that specific purpose. Bubenko jr. et al. (2001) explains it as follows:

In Enterprise Modelling (EM) we focus on modelling an enterprise. Typical examples of enterprises are private companies, public authorities or bodies, academic institutions or other organisations. There are different views of EM. Furthermore, Enterprise Modelling is a structured technique to describe different aspects of an enterprise. (p. 5).

Furthermore, the authors explain Enterprise Modeling as follows:

EM is a process where an integrated and negotiated Enterprise Model is created, describing a specific enterprise from several different perspectives. The perspectives vary, depending on the focus of the EM method and the problem that is being addressed using the method. Examples of such perspectives are processes, business rules, concepts/information/data, goals, actors, and requirements. (Bubenko jr. et al. 2001 p. 6)

Enterprise Modeling can also be described as a generalization and extension of system analysis and design, modeling the pragmatic aspects, the semantic aspects and syntactic elements of an information system (Gustas & Gustienė 2004). By applying Enterprise Modeling, focus is put on modeling (designing) and integrating the business processes of the enterprise in focus (Vernadat 1996).

The pragmatic aspects of EM are represented by a set of pragmatic dependencies used to illustrate goals, problems, and opportunities together with positive and negative influences. Since the pragmatic part is outside the scope of this thesis it will not be discussed further. Readers interested can read more about the pragmatic aspects and pragmatic dependencies in Gustas (1998) and Gustas & Gustienė (2004, 2008).
In this thesis, focus is put on the semantic aspects. In EM, the semantic aspect is represented by both a set of static dependencies and a set of dynamic dependencies. Static dependencies (Figure 5) are used to illustrate the static part of an information system, meaning both what data are stored in the database and what data the information system needs for processing the functionality it should provide.

![Figure 5](image)

**Figure 5.** Representation of static dependencies in EM (adapted and modified from Gustas & Gustienė (2004))

Dynamic dependencies are divided into state dependencies and communication dependencies (Figure 6). State dependencies are used to illustrate state changes together with pre- and post-conditions for actions. Communication dependencies are used to illustrate relations between actors, their actions and communication flows.
Finally, the syntactic aspects are represented by a set of syntactic elements (Figure 7).

Syntactic elements are considered as CASE tool-dependent (Gustas & Gustiené 2004; Jakobsson & Gustas 2004) and therefore the list is not exhaustive, meaning elements that are needed for a specific task can be added on demand.

The main goal while applying the Enterprise Modeling approach is to end up with a consistent, coherent and complete specification of an information system (Gustas & Gustiené 2004) using one schema type (Gustas & Jakobsson 2004). This is an important issue, since studies have indicated that using one schema type may result in less confusion for the users than keeping the static and dynamic schemata separate (Turetken & Shuff 2002; Turetken et al. 2004).

To summarize, applying the notation given in EM during conceptual modeling has several advantages compared to more traditional modeling approaches. First, in EM, every static concept is modeled as a box. This reduces the problems (similarities and differences) that might occur in schema integration.
This means that while conducting conceptual database design (modeling the static part) a concept is the only thing that is given a name. Second, in EM a dependency between two static concepts is classified as association, composition, aggregation, specialization, generalization or instance-of (see Figure 5). Third, by applying EM, designers and users can represent, describe and illustrate both the static and the dynamic aspects using one schema type. Forth, by applying EM, it is possible to illustrate and represent the system analysis patterns: sequence, synchronization, iteration, selection and search (Gustas & Gustiené, 2009). In Figure 8 both the static and the dynamic aspects of an information system are integrated into one schema. Among other things, this makes it possible to check that the use of concept names (labels) is consistent in both the static and the dynamic schemata. The primitives used in Figure 8 are described in Figure 5 and Figure 6.

![Figure 8. Example of integrated schema including both static and dynamic aspects (adapted and modified from Gustas (2005))](image)

The static aspects of Figure 8 can be described as follows: Person is the main static concept that is described and illustrated. Person has three associated concepts Surname, Name and Civic Registration Number. Person [Not Employed], synonymous with Applicant, is a specialization of Person and at the same time the post-condition of action Apply and a pre-condition of both actions Reject and Employ. Person [Employed], synonymous with Employee, is also a specialization of
Person and a post-condition of action Employ. Person \([\text{Employed}]\) also has an additional concept associated, named Date of Employment.

The dynamic aspects of Figure 8 can be described as follows: Person is the actor initiating the interaction loop by sending the flow Application through action Apply to the receiving actor Company. Company is then responsible to reply. The post-condition of the action Apply is Person \([\text{Not Employed}]\) which can be viewed as an intermediate step since Company has not yet decided whether to Reject or Employ Person \([\text{Not Employed}]\). Actor Company can respond by either rejecting the Application (Reject) or accepting the Application (Employ). If Company decides to Reject the Application received from Person then the flow back to Person is the same Application. However, if Company instead decides to Employ the Applicant (Person \([\text{Not Employed}]\)) then the flow back to Employee, illustrated as a specialization of Person, is instead a Contract. If that is the case then Person \([\text{Not Employed}]\) is re-classified to Person \([\text{Employed}]\) (Employee). Finally, the post-condition of action Apply can also be described as a creation action and the missing post-condition of action Reject as a termination action (Gustas & Gustiené 2009).

It is important to point out that by applying the Enterprise Modeling approach, a concept can be interpreted differently depending on the concept and dependency in focus (Gustas & Gustiené 2002).

3.1.5 Summary of Modeling Languages and Approaches

Today, several modeling languages and approaches are used while conducting not only requirement elicitation but also requirements engineering. Although many of the used modeling languages and approaches tend to focus on technical parts and/or the implementation level of the future information system (Gustas & Gustiené 2002), this is not always desirable.

In this thesis I have focused on not only the integration of static schemata but also on the integration of static and dynamic schemata. In doing so, the chosen and applied notation should focus on the early phases of information systems development, rather than the later phases focusing on the technical parts and the implementation level. By applying the chosen notation, it should also be possible to integrate the static and the dynamic schemata into one schema type. This means that neither the Entity-Relationship modeling language nor the Unified Modeling Language fits the criteria. The Entity-Relationship modeling
language can be used to represent, describe and illustrate (model) the static aspects of the future information system, meaning it can be used to describe and illustrate the future database from an implementation dependent perspective. The Unified Modeling Language on the other hand can be used to describe not only the static but also the dynamic aspects of an information system. Although similar to the Entity-Relationship modeling language, the Unified Modeling Language is implementation dependent developed to illustrated and describe (model) issues close to implementation.

The Klagenfurt Conceptual Pre-Design Model fulfills the purpose concerning implementation independence. However, by applying the Klagenfurt Conceptual Pre-Design Model, it is not possible, without the developed NIB-A tools, to integrate the static and the dynamic schemata into one schema type. Therefore, the only modeling language that fulfills the criteria mentioned above is the Enterprise Modeling approach which is also, in this thesis, the applied modeling language and notation. However, it should be noted that in three of the papers included in this thesis (Bellström et al. (2007, 2008, 2009)) a first attempt to identify similarities between the Enterprise Modeling approach and the Klagenfurt Conceptual Pre-Design Model is conducted. This means that constitutes used in the modeling languages have been reduced to a minimal for modeling static as well as for dynamic aspects of a database and information system.

3.2 Integration Process

Schema integration is the process of integrating several source schemata, also referred to as views, into one global schema representing the domain and enterprise of interest. Several explanations and definitions on schema integration have been proposed in the literature (see Batini et al. 1986, 1992; Boman et al. 1997). Despite the differences between these definitions, the main application of schema integration is still to integrate source schemata (the parts) into one schema (the whole). In this section, three integration processes, integration of static schemata, integration of dynamic schemata and integration of static and dynamic schemata, are described and discussed.

3.2.1 Integration of Static Schemata

Several methods and approaches for the integration of static schemata have been proposed. In Batini et al. (1986), the authors concluded that an integration process is comprised of, or at least a mixture of, the following four phases: pre-
integration, comparison of the schemata, conforming the schemata, and merging and restructuring (Figure 9).

![Figure 9. Process of Integration of Static Schemata (adapted and modified from Bellström (2006a p. 20))](image)

Arrows moving from left to right illustrate and indicate feed-forward and arrows moving from right to left illustrate and indicate feed-back. The arrows also indicate that the schema integration process is not strictly sequential but rather more iterative moving back and forth between the phases. The integration process proposed and described by Batini et al. (1986) has influenced many integration methods, not only using all phases (e.g. Shoval & Zohn 1991; Theory 1999) but also using parts of it (Dupont 1994; Spaccapietra & Parent 1994; Lee & Ling 2003). Finally, extension of the integration process with an additional phase has also been suggested (Bellström 2006b). In this section a summary of each phase is given. A longer and more exhaustive description and discussion regarding the schema integration process for static schemata is given in Bellström (2006a).

As stated by Song (1995), during pre-integration, at least three tasks should be carried out. These are: translate all schemata into a canonical modeling language, check for conflicts and inconsistencies in each schema and select integration strategies.

In comparison of the schemata at least three other tasks should be carried out. These are: identification of name conflicts, identification of structural conflicts and finally identification of inter-schema properties (Johannesson 1993). It should be noted that comparison of the schemata has been described as not only an important (Song 1995) but also as a most difficult phase (Ekenberg & Johannesson 1996; Lee & Ling 2003) in schema integration.
In conforming the schemata the similarities, e.g. synonyms and hypernyms-hyponyms, and differences, e.g. homonyms, between two source schemata identified in the previous phase are resolved. Also this phase has been pointed out as critical (Lee & Ling 2003) and as a key issue (Spaccapietra & Parent 1994) in schema integration. While conforming the schemata, the users not only analyze how to name concepts but also how to change the structure of the schema including both concepts and dependencies. Changing the structure should be done without losing any of the used concept names and without losing any of the dependencies that represents an important dependency between two concepts.

Finally, in merging and restructuring, the schemata are superimposed, merged, into one global schema and later restructured, aiming to get rid of redundant concepts and dependencies. In Lee and Ling (2003), the authors propose that the database administrator has to decide what relationships are redundant and therefore removable. However, it is important to point out that it is seldom clear if a concept and/or dependency is redundant, e.g. could be derived from other concepts or dependencies, since concepts and dependencies may have different meanings for different users (Day et al. 1999). In other words, while restructuring the global schema, it is important that designers and users only remove concepts and/or dependencies that are truly redundant otherwise they might instead cause several problems such as semantic loss.

3.2.3 Integration of Dynamic Schemata

Integration of dynamic schemata is more complex compared to integration of static schemata. Not only because several more modeling constituents are introduced but also because schemata illustrating the dynamic aspects do not follow similar principles to the ones used for schemata illustrating the static aspects of an information system. Principles for representing the static part of a domain or enterprise are as follows: objects can be classified to classes or entity types. Classes and entity types are related (connected) to each other by relationships. In addition both classes and entity types might have attributes. One approach proposed for integration state charts is given in Frank and Eder (1999). The authors propose the following four phases: schema analysis, schema transformation, schema merging, and finally schema restructuring. However, in this thesis, focus is put on integration on a higher level of abstraction compared to the one demonstrated by Frank and Eder (1999). Therefore, the integration process described below is discussed in detail. It should be noted that in the
described integration process (see Bellström et al. (2009)), user participation is emphasized since the schemata focus on what the users want the information system to perform rather than being a description of the implementation. Another aspect worth mentioning is that the schemata, prepared to be merged, only contain notations for conditions and actions (in Bellström et al. (2009) we named these process-types) which have been mentioned as the minimal constituents for modeling the dynamic aspects of an information system (Fiedl et al. 2003; Kop & Mayr 2002). In the rest of this section, the four phases proposed in Bellström et al. (2009) are described. The phases are: preparation of source schemata, recognition of conflicts and commonalities between source schemata, resolution of conflicts and commonalities between source schemata, and finally merging the source schemata and restructuring the global schema (Figure 10).

**Figure 10. Process of Integration of Dynamic Schemata**

In preparation of source schemata, the source schemata are prepared for comparison. This means that the used terms (e.g. concept names) are checked against a number of naming guidelines (a controlled subset of natural language). One goal is to ensure that meta-information about the static concepts is made explicit via the concept name. Making meta-information about the static concepts explicit via the concept names is of specific importance for the described integration process since only conditions together with actions are used in the schemata. Besides the naming guidelines, focus is on recognizing and resolving similarities, e.g. synonyms, and differences, e.g. homonyms, within one schema. These problems have to be resolved before moving on to the second phase.

In recognition of conflicts and commonalities between source schemata, two schemata are compared. Binary integration (Batini et al. 1986, 1992) is used while trying to recognize similarities and differences between two source schemata. It is possible to manually carry this out; however, it is recommended to use
computer support to facilitate the step. Domain repositories are one type of computer support that might be used. It is recommended that both linguistic comparisons, comparing the concept names, and structural comparisons, comparing the concepts’ neighborhoods, are used. The motivation for this is that by doing so it is possible to identify both similarities and differences between two source schemata that otherwise might pass unnoticed.

In resolution of conflicts and commonalities between source schemata, the differences and similarities identified between two source schemata are resolved. Two specific types of problems that might problematize the integration process of dynamic schemata is condition overlapping and process type overlapping (Bellström et al. 2008, 2009). Overlapping means that parts of one condition/process within one schema, describing the dynamic aspects of an information system, correspond partially to a condition/process within another schema. In this step, it is important to involve the users to ensure correct integration and to ensure that neither conditions nor process types are lost which often cause semantic loss.

Finally, in merging the source schemata and restructuring the global schema, the schemata are superimposed into one schema and later on also restructured. Here it is again important to emphasize that during this phase neither conditions nor process types should be lost, otherwise we might cause semantic loss. In addition, it is important that the domain experts, the users, are again involved to ensure correct integration. If it is needed, domain repositories might also be used to facilitate the integration process.

3.2.3 Integration of Static and Dynamic Schemata

Following the traditional database design method, schema integration can be performed at the conceptual level, the logical level or the physical level. During schema integration, focus can also be placed on different schema types such as schemata representing, describing and illustrating the static and/or the dynamic aspects of an information system. In this thesis, it is argued that schema integration should be performed not only focusing on the static schemata or the dynamic schemata but also focusing on integrating the static schemata and dynamic schemata into one global schema. In Bellström and Jakobsson (2006), we studied the problem from both a broader perspective and a more narrow perspective and in this section the broader perspective will be addressed.
Following the proposed method (see Figure 11), it is possible to first integrate the static schemata and, in a second step, to integrate the dynamic schemata. Finally, in a third step, we check for inconsistencies within the global integrated schema comprised of both static and dynamic aspects. The proposed integration method is independent of modeling language/approach and therefore applicable on any modeling language/approach as long as the chosen modeling language supports the possibility to first design the static schemata and later on continue with the dynamic schemata. By applying the proposed integration method, the chosen modeling language should also support the design of both the static and the dynamic aspects of an information system. This means that it is possible to use the most fitting and specific integration method (approach) at hand.

![Diagram of integration process](image)

**Figure 11.** Method for integrating static and dynamic schemata (adapted and modified from Bellström & Jakobsson (2006 p. 636))

### 3.2.4 Summary of Integration Approaches

Schema integration can be performed on different levels of abstraction with different schema types in focus. In 3.2.1, integration of static schemata was presented. In 3.2.2 an overview of integration of dynamic schemata, focusing
on actions (process-types) and conditions was given. Finally, in 3.2.3 an overview of integration of static and dynamic schemata, focusing on integrating both the static and the dynamic aspects of an information system, was given. It should be noted that the proposed method for integration of static and dynamic schemata could include syntactic elements (e.g. database, database management system and software component) illustrating some logical parts of the described information system.
4 Maintaining Vocabulary in Schema Integration

The purpose of this chapter is to present a summary of the first research topic. This is achieved by addressing the motivation, problem, approach and solution and finally contribution to the field. In the first section the research topic is introduced and motivated. In the second section the solution to the addressed problems is presented. Finally, in the third section the contributions found in this research topic are presented.

4.1 Motivation

The problems of conceptual schema integration are manifold in which the two major problems originate from structural and semantical similarities and differences between two source schemata. Batini et al. (1986) describe the problems as follows:

The basic problems to be dealt with during integration come from structural and semantical diversities of schemas to be merged. (p. 329)

Both semantic similarities and differences are often difficult to recognize and resolve. Therefore it is very important to understand the intended meaning of a concept (object). Song (1995) expresses it as follows:

In the study of schema integration, the major problems lie in understanding and identifying the intended meaning of objects in different schemas, i.e. the schema object semantics, which are typically represented by words. (p. 34).

This is also emphasized in Ambrosio et al. (1997) where the authors point out that during schema integration:

[…] schema semantics is fundamental, and must be understood each time schema manipulation is performed. (p. 126).

However, while trying to understand and identify the intended meaning of concepts, users and designers might make wrong decisions or even try to oversimplify the source schemata and/or the integrated schema. In this chapter, several consequences of such wrong decisions and/or simplification originating from the vocabulary used in the enterprise in focus are dealt with. In this chapter, a schema is mainly treated as a communication “tool” (cf. Ambrosio et al. 1997; Boman et al., 1997; Hoppenbrouwers et al. 2005b) between users and designers. Focus is put on the static part of conceptual schema integration in conceptual database design (see Figure 1).
The main problem treated in this chapter, *semantic loss*, originates from the endeavor to minimize the number of concepts and their relationships in the integrated conceptual schema. Batini et al. (1992) describe *minimality* as follows:

A schema is **minimal** when every aspect of the requirements appears only once in the schema. We can also say that a schema is minimal if no concept can be deleted from the schema without losing some information. (p. 140)

It should be noted that the description given in Batini et al. (1992) focuses on implementation details and not conceptual details. In this thesis, it is important to point out that it is schemata with focus on implementation that should aim to be minimal. In other words during conceptual schema integration, concepts and dependencies should be retained as long as possible to avoid semantic loss. A dependency should be interpreted as a link or connection between two concepts.

Problems related to semantic loss and information loss have also been discussed in different contexts such as resolution of structural conflicts (Lee & Ling 2003; Spaccapietra & Parent 1994) and derived information (Dey et al. 1999; Rauh & Stickel 1993). Nevertheless, the moment at which a concept and/or dependency may be deleted still needs to be analyzed and discussed in depth. On the other hand, depending on the chosen perspectives, striving towards minimizing the number of concepts and dependencies could cause problems such as *information loss* (Lee & Ling 2003), *constraint loss* (Lee & Ling 2003), *concept name compression* (Bellström & Carlsson 2004, 2006) and *semantic loss* (Bellström 2009).

From that short introduction and motivation, the papers referred to as Bellström & Carlsson (2004, 2006) and Bellström (2009) are placed within the first research topic: *Maintaining Vocabulary in Schema Integration*.

### 4.2 Problem, Approach and Solution

The main problem of this research topic, referred to as *semantic loss*, originates from the endeavor to minimize the number of concepts (including their names), and/or dependencies in the global conceptual database schema. Minimizing the number of concepts and/or dependencies might result in problems connected to vocabulary and difficulties in understanding the meaning and the contents of the integrated conceptual database schema. In
general, it might cause problems when users are verifying or declining the integrated conceptual schema since one or several concepts and/or dependencies are lost.

Finally, a comment regarding minimization, semantic loss and redundancy is required. While discussing these problems, focus is put on semantic integration and not on syntactic. This means that semantic loss is undesirable but so is redundancy. Therefore minimization should be conducted without causing semantic loss (Bellström 2009) or information and/or constraint loss which is not always the case (Lee and Ling 2003).

4.2.1 Concept Name Compression

In the first paper of this research topic (Bellström & Carlsson 2004) we studied what happens if one concept name is used for several concepts, which is often the case while trying to resolve synonyms. In the study, the problem became concept name compression and we criticized the semantic multitude that is often concealed in the integrated conceptual schema. In Figure 12 and Figure 13 the problem of concept name compression is illustrated, meaning after schema integration the two concept names (labels) are compressed into one name (label). The primitives used in Figure 12 and Figure 13 are described in Figure 5.

Figure 12. Two synonymic concept names have been recognized
Compressing (merging) two concepts names into one also means that Customer is used for both Customer and Client but Client will still exist in the shadow of Customer; it might even be used by users while discussing the integrated global schema.

More precisely this meant that in the paper, the problem of semantic loss was studied from a slightly different perspective compared with the one adopted in Lee and Ling (2003). Lee and Ling (2003) describe the problems of information loss and constraint loss in connection with schema transformations while using the terms information preserving and constraint preserving. They put it as follows:

Informally, a transformation is information preserving if any database instance structured according to the original schema can be losslessly converted into a database instance according to the transformed schema, and vice versa, while a transformation is constraint preserving if the constraints expressed in the original schema can also be expressed in the transformed schema. (p. 238)

In the paper, we focused on synonyms, often referred to as a name conflict (Batini et al. 1986; Lee & Ling 2003; Spaccapietra & Parent 1994) or a semantic conflict (Song 1995). Synonyms occur if two concept names (labels) are used for one concept. See for instance Figure 13 where Customer and Client are used to represent the same concept. The problem of concept name compression might occur if the users together with the designers decide to rename one or both concepts (see for example Batini & Lenzerini 1984; Batini et al. 1986; Johannesson 1993). Renaming is mentioned in the literature as the most common resolution technique for synonyms and homonyms (similarities and differences). Using renaming means that one name is chosen to represent all the other concepts that have been identified as synonymous. This could mean that either the first concept is renamed or the second concept is renamed or both concepts are renamed. If renaming is used as a resolution technique, two
concept names (labels) are compressed into one concept name (label). Compressing concept names not only causes concept name compression but also impoverishes the vocabulary used in the integrated conceptual database schema. It might even impoverish the meaning of the integrated conceptual database schema since one or several parts are changed or lost. The problem of concept name compression was summarized in Bellström (2006a) and defined as:

a state which may occur if several concept names are merged (compressed) into one concept name, for example when choosing one of the concept names to represent a concept when trying to resolve a synonym conflict. (p. 46)

To summarize, the main problem in the first paper addresses what happens when two concept names are compressed (merged) into one and what the consequences of such an act might be in schema integration.

The proposed solution for the addressed problems is to increase the possibility for maintaining the vocabulary used in the conceptual database schemata during schema integration. This means retaining the concept names (labels) in the schema integration process and at the same time enabling mutual interpretation and understandability of the integrated conceptual database schemata. If two concept names have been compressed, we proposed two generic methods on how to enrich the vocabulary used in the integrated conceptual database schema by reconstructing the concept names.

In Bellström and Carlsson (2004) we demonstrated two generic methods on how to reconstruct mutual understanding of concept names used in a database and/or an integrated conceptual database schema and how to obtain, support and maintain intersubjectivity of concept names used in a database and/or an integrated conceptual database schema. It should be noted that the proposed methods are generic in the sense that they are applicable not only for database schemata but also for information systems schemata.

In the first proposed method, reconstruction of mutual understanding of concept names, focus was put on catching the users pre-understanding of the meaning of the concept names (labels) in a database and/or an integrated conceptual database schema. By catching users' pre-understanding, we are also able to catch how users define concepts and if they actually use other concept names (labels) than the ones defined in the database and/or the integrated
conceptual database schema. The method is based on a set of questions the
designer asks the users. The questions are inspired by Heidegger (1981a, 1981b)
and are for instance: by whom, in what purpose, to whom, to what, in what context and
why.

In the second proposed method, obtaining, supporting and maintaining
intersubjectivity of concept names, users should, during a modeling session
(seminar), simulate how the concept names (labels), identified while applying
the first proposed method, are used in a daily working situation. During the
modeling session, users should also clearly explain and define the concepts and
what names (labels) they use. The proposed method was inspired by an
approach for supporting users’ comprehensive learning proposed by Eriksson
et al. (1988). We proposed a method comprised of the following three steps:
preparation, simulation and follow-up. The core is in the simulation phase, where
concept names (labels) in a database and/or an integrated conceptual database
schema are in focus. The preparation phase contains user training regarding the
used modeling language and its primitives. The simulation phase contains
simulations of the concept names during a modeling seminar. Finally, the follow-
up phase contains a follow-up study regarding the results of the simulation. In
other words, the proposed method is comprised of three phases including a
planning phase, a realization phase and a follow-up phase.

4.2.2 Revisiting Concept Name Compression

In the second paper of this research topic (Bellström & Carlsson 2006), we
extended the discussion regarding concept name compression and included
resolution techniques for homonyms and inter-schema hypernym-hyponym
properties. The motivation for the second paper was to try to resolve not only
similarities between two source schemata, e.g. synonyms and inter-schema
hypernym-hyponym properties, but also differences between two source
schemata, e.g. homonyms, where the risk and danger of causing concept name
compression is high. Homonyms occur if one concept name (label) is used to
represent two or more concepts (labels) such as the use of Size in both the
context of TV and the context of Product as illustrated in Figure 14.
An inter-schema hypernym-hyponym property occurs when one concept name (label) is defined as a specialization of another concept name (label) and/or the first concept name (label) is defined as a generalization of the other concept name (label). For instance, using *Order* in one schema and *High Priority Order* in another as illustrated in Figure 15.

This means that an inter-schema hypernym-hyponym property indicates that concept names are similar but not the same. This is very different compared to synonyms which indicate that two concept names represent the same concept.

Finally, we addressed the question of what happens to the possibility of reaching mutual interpretation of a schema if concept name compression occurs.

To summarize, the main problem studied in Bellström and Carlsson (2006) concerned how to resolve not only similarities, such as synonyms and inter-schema hypernym-hyponym properties, but also differences, such as homonyms, between two source schemata without causing semantic loss.

In the paper, we presented concrete resolution methods on how to, between two source schemata, resolve synonyms and inter-schema hypernym-hyponym properties (Figure 16) as well as homonyms (Figure 17) without losing any
concepts and/or dependencies. More precisely, this meant methods on how to resolve both similarities and differences between two source schemata without losing any of the original concept names and/or dependencies.

![Diagram](image1)

**Figure 16.** Resolution of inter-schema hypernym-hyponym property applying inheritance dependency

![Diagram](image2)

**Figure 17.** Resolution of homonyms applying sharper cardinality, inheritance dependency and new concepts (TV.Size and Product.Size)

It should be noted that in resolving similarities and differences between two source schemata, a dependency might transform and/or a new dependency might be introduced to sharpen the meaning of a schema. In the proposed resolution techniques, the importance of retaining concept names during the integration process was also emphasized.

### 4.2.3 Semantic Loss

In the last publication (Bellström 2009) placed within this research topic the problem of semantic loss was studied from broader perspective. More precisely this meant that not only the problem of *semantic loss* was studied, but also the problem of *information loss* and the problem of *concept name compression*. Three conceptual modeling issues that might cause semantic loss in the integration process were also studied: *minimality, the use of different model constructs and the use of non-comparable transformations*. In Bellström (2009) an overall picture of what the problem of semantic loss actually means in connection with the vocabulary
used in the source schemata in the integration process was given. Furthermore, resolution methods on how to avoid semantic loss were proposed, demonstrated and compared with other proposed resolution methods.

To summarize, the overall problem studied in Bellström (2009) was how to avoid semantic loss occurring while conducting schema integration in conceptual database design.

In Bellström (2009), I studied how the problem of semantic loss affects all the phases in the schema integration process. For that purpose, I studied resolution techniques for similarities such as synonyms and inter-schema properties (the hypernym-hyponym property and the holonym-meronym property) and differences such as homonyms. An inter-schema holonym-meronym property occurs if one concept is defined as a part of another concept, meaning two or more concepts are either composed of or aggregated into one concept. For instance, Order is composed of one or several OrderLine(s) (see Figure 16). At the same time, this indicates that one specific OrderLine is a part-of a specific Order. To resolve the mentioned inter-schema properties without causing semantic loss, new or sharper dependencies need to be introduced between the concepts such as inheritance or aggregation/composition (see Figure 16). To improve schema integration as such, the analysis demonstrated how the issues of semantic loss, concept name compression and information loss relates and/or differentiates.

In the paper, I analyzed and demonstrated how the three conceptual modeling issues of minimality, the use of different modeling constructs and the use of non-comparable transformations could cause semantic loss and how it could be avoided in the schema integration process. In the paper, the proposed solution to the problem of minimality was to retain concepts and dependencies in the schema integration process as long as possible. It was also argued that a modeling language suitable for conceptual database design should be applied. More precisely this meant that the chosen modeling language should not differentiate between entities (objects) and attributes but in instead focus on only two modeling constituents: concepts and dependencies (relations between concepts). Finally, so called non-comparable transformations, such as renaming concepts, should be avoided. Alternative resolution techniques to the mentioned problems were also proposed and demonstrated.
4.3 Contribution

There are two main contributions given in the first research topic; Maintaining Vocabulary in Schema Integration. The first is given in the analysis on what semantic loss is in connection with schema integration. In the first contribution, distinguishing between the three related issues of concept name compression, information loss and semantic loss is included. This contribution also includes an analysis and demonstration of the problems and why they occur. Figure 18 illustrates how the three problems are related.

Figure 18. Graphical description of Information Loss, Concept Name Compression and Semantic Loss

Figure 18 should be interpreted as follows: information loss is caused either by loss of concept or loss of dependency. This means that either a concept, including its name, or a dependency is deleted (lost) from the schema. Information loss is a problem that has to be dealt with while working on an implementation dependent level. The motivation for this is that the implementation dependent level deals with the ability to store a real-world fact in the future implemented database. Information loss is in this thesis defined as:

a problem that occurs if one or several concepts and/or dependencies that make it possible to store a real-world fact in the future database are lost in the schema integration process.

Furthermore, concept name compression and semantic loss have been defined as follows:
a state which may occur if several concept names are merged (compressed) into one concept name, for example when choosing one of the concept names to represent a concept when trying to resolve a synonym conflict. (Bellström 2006a p. 46)

**Semantic loss** is [...] a problem that occurs if one or several concept names and/or dependencies describing the meaning of a concept are lost during the view integration process. (Bellström 2009 p. 965)

The usage of “view” in the definition of semantic loss should be treated as synonymous with the usage of “schema”.

On the *implementation independent level* (see Figure 18) focusing on describing the meaning of a concept, two additional problems have to be dealt with: concept name compression and the use of a less strict dependency. Nevertheless, all four problems shown at the implementation independent level actually cause semantic loss occurring. This is the case since concept name compression actually hides one or several concept names (labels) behind one concept name (label). If a less strict dependency is used, for instance choosing association instead of inheritance, we lose semantics and precision in the schema. Losing or deleting either a concept and its name or a dependency also causes semantic loss, since we lose a concept and/or dependency that is used while describing the meaning of a concept.

The *second* contribution is given in the proposed methods on how to resolve similarities and differences between two source schemata without semantic loss occurring. Resolution techniques for synonyms and homonyms also referred to as name conflicts (cf. Batini et al. 1986; Lee & Ling 2003; Spaccapietra & Parent 1994) and the inter-schema hypernym-hyponym and holonym-meronym properties (e.g. Lee & Ling 2003; Storey, 1993, 2001) have been proposed and demonstrated. It should be noted that the proposed resolution methods do not only counter an impoverishment of the vocabulary used in the source schemata but also counter the occurrence of concept name compression and semantic loss (see for instance Figure 19). This means both maintaining the vocabulary used in the schemata and retaining the meaning of the schemata during the schema integration process. Figure 19 illustrates the proposed resolution technique for two synonymous concepts. As illustrated in Figure 19, both concepts causing the synonym conflict are retained even after the problem has been resolved.
The resolution technique proposed and applied in Figure 19 is known as mutual inheritance dependency (Gustas 1997). The resolution technique is in line with the following statement given by Martin and Odell (1998): 

While standardization is useful, enforcing the one-and-only–one name is impractical in all situations. (p. 19).

To illustrate and complement the second contribution given in this research topic, a more comprehensive study regarding resolution techniques was also conducted by Bellström (2009). In that study, all proposed resolution techniques on how to counter semantic loss were illustrated and discussed. Note that since both papers referenced as Bellström and Carlsson (2006) and Bellström (2009) have contributions to research topic one, Maintaining Vocabulary in Schema Integration, and research topic two, Integration of Static Schemata, contributions given in the research topics are overlapping. This is natural since parts of the published papers belong to research topic one and parts belong to research topic two.

### 4.4 Contribution and Evaluation in Relation to Design Science

In this section, research contribution and evaluation method in relation to design science is addressed. As pointed out earlier in design science research, the IT-artifact is in focus. An artifact is classified as a construct, a model, a method or an instantiation (Hevner et al. 2004; March & Smith 1995). The first contribution of research topic one is classified as a construct, meaning, it contributes: 

[…] to describe problems within the domain and to specify their solutions. (March & Smith 1995 p. 256)
The contribution was evaluated applying a descriptive evaluation method: *informed arguments*. In this case, a descriptive evaluation method means the use of relevant research results:

> [...] to build a convincing argument for the artifact’s utility (Hevner et al. 2004 p. 86)

The descriptive evaluation method was chosen since in this thesis the problem of semantic loss was studied from a new perspective: how to maintain the vocabulary in the source schemata. Having that focus, the vocabulary in the source schemata is retained and an impoverishment of the vocabulary is avoided.

The second contribution of research topic one is classified as *method*, meaning it describes:

> [...] a set of steps [...] used to perform a task. (March & Smith 1995 p. 257).

The second contribution was evaluated using one analytical method: *static analysis* and two descriptive methods: *informed arguments* and *scenarios*. Hevner et al. (2004) describe static analysis and scenarios as follows:

**Static Analysis**: Examine structure of artifact for static qualities (e.g., complexity) [...]  

**Scenarios**: Construct detailed scenarios around the artifact to demonstrate its utility (p. 86).

In this case ‘static analysis’ means examining how well the proposed resolution methods works while focusing on the ability to retain concept names and/or dependencies. Retaining a dependency (a connection/link between two concepts) might also mean introducing a sharper one. Finally, ‘scenarios’ means that the artifact, the resolution methods, has been tested and applied (see for instance Bellström (2009)) to demonstrate its utility.
5 Integration of Static Schemata

The purpose of this chapter is to present a summary of the second research topic. This is achieved by addressing the motivation, problem, approach and solution and finally contribution to the field. In the first section the research topic is introduced and motivated. In the second section the solution to the addressed problems is presented. Finally, in the third section the contributions found in this research topic are presented.

5.1 Motivation

Both similarities and differences arise between two source schemata because users have different references and use different vocabulary. A general description of the term conflict is given by Batini et al. (1986), in which $R_1$ and $R_2$ are abbreviations for representation one and representation two, as follows:

A conflict between two representations $R_1$ and $R_2$ of the same concept is every situation that gives rise to the representation $R_1$ and $R_2$ not being identical. (p. 335).

Johannesson (1993) describes and explains the origin for some of the conflicts that might occur in the following way:

Fundamental to most problems with schema integration is that a Universe of Discourse can be modelled in many different ways. The same phenomenon may be seen from different levels of abstraction, or represented using different properties. Different terms can denote the same concept, and different modelling structures can represent the same reality. (p. 10).

Therefore, while conducting schema integration we have to recognize and resolve not only differences but also similarities between two source schemata. Differences and similarities between two source schemata are identified during comparison of the schemata. Comparison of the schemata has been called the most important (Song 1995), difficult (Ekenberg & Johannesson 1996) and challenging (Johannesson 1993) phase in schema integration. While conforming the schemata, the differences and similarities identified in the previous phase are resolved. Conforming the schemata has also been mentioned as a critical issue (Lee & Ling 2003) and a key issue (Spaccapietra & Parent 1994) in schema integration. Together these two phases constitute a cornerstone in the schema integration process. The challenges, difficulties and issues expressed above indicate its complexity.
As can be seen from the referenced papers presented below, papers Bellström and Carlsson (2006) and Bellström (2009) are also included in the first research topic *Maintaining Vocabulary in Schema Integration*. This overlap is necessary since both papers are bridging the two research topics. This means that parts of the papers are placed within *Maintaining Vocabulary in Schema Integration* and parts are placed within *Integration of Static Schemata*. For instance, in Bellström (2009) the description regarding concept name compression belongs to the first research topic while the proposed and illustrated resolution methods belong to both the first research topic and the second research topic.

In this chapter, schema integration in conceptual database design (Figure 1) is emphasized. This means manual integration of static schemata, applying the notation used in the Enterprise Modeling approach (Gustas & Gustiené 2004). Focus is mainly on how to deal with issues in connection with comparing and conforming the schemata in the schema integration process, more specifically, the recognition and resolution of similarities and differences between two source schemata. Comparing two source schemata is always assumed since the proposed approach applies binary integration (see Batini et al. (1986, 1992)). However, it should be noted that similar problems, as discussed below, could arise while checking for differences and similarities within one schema (intra-schema problems) during pre-integration, in the schema integration process. Nevertheless, intra-schema problems are not treated in this chapter since focus is put on problems while comparing two schemata, resolving problems (inter-schema conflicts and inter-schema properties) between two source schemata and integrating two source schemata. Finally, in this chapter a schema is not only viewed as a communication “tool” but also as a first draft, on a conceptual level, on how the future database should be designed.

From that short introduction and motivation, the papers referred to as Bellström (2005, 2006b, 2009), Bellström & Carlsson (2006) and Bellström et al. (2007) are place within the second research topic: *Integration of Static Schemata*.

### 5.2 Problem, Approach and Solution

The two main problems studied in this chapter are recognition and resolution of differences and similarities between two source schemata in the process of integrating static schemata. Many of the problems connected to integration of static schemata originate from the use of different vocabulary and/or the use of incomplete concept names. For instance Batini et al. (1992) point out that the
risk for a homonym conflict is higher when the vocabulary of terms is small. Similar reasoning is given in connection with schema integration on the logical level where the same problem occurs in schema integration on the conceptual level, as Kim and Seo (1991) point out that the use of incomplete concept names could cause homonyms. In general, using incomplete and/or imprecise concept names might cause several problems. The most common resolution method for homonyms and synonyms given in the literature is renaming. Renaming might, however, cause several other problems such as concept name compression and semantic loss.

Therefore, in this chapter, we study the capability to identify methods that make it possible to recognize not only differences, such as homonyms, but also similarities, such as inter-schema properties, between two source schemata. The possibility to identify methods that resolve the recognized differences and similarities between two source schemata, without causing concept name compression and/or semantic loss, are also studied. More precisely, this means resolution methods that do not delete but instead retain concepts, including their names, and at the same time counter an impoverishment of the vocabulary used in the schemata.

5.2.1 Recognition and Resolution of Homonyms

In the first paper of this research topic (Bellström 2005) both problems of recognizing and resolving homonyms in schema integration were addressed. The instance-of dependency was of specific interest since, including it in the conceptual schema, several new opportunities, challenges and problems have to be addressed. One such problem is how to resolve similarities and differences between two source schemata including the instance-of dependency. In He and Ling (2004) the authors addressed the problem in connection with the use of Entity-Relationship diagrams and called it schematic discrepancy. In the paper the authors described schematic discrepancy as a problem that occurs when

\[\ldots\text{the same information is modeled as data in one database, but metadata in another. } \text{(He \\& Ling 2004 p. 245).} \]

Including the instance-of dependency also means that we are able to integrate several levels such as the model-level and the instance-level in one schema. To demonstrate and clarify how this is supported in this thesis Figure 20 is included. In Figure 20 TV is both an Item since TV inherits Item and a Product
since $TV$ is an instance-of $Product$. The primitives used in Figure 20 are described in Figure 5.

![Diagram of inheritance and instance-of dependencies in one schema](image)

**Figure 20. Inheritance and instance-of dependencies in one schema**

To summarize, the problems studied in Bellström (2005) were a) how to manually recognize and b) how to manually resolve homonyms in schema integration while applying the notation of the Enterprise Modeling approach.

The proposed solution to the first problem was a two-step recognition method named the dependency approach. The dependency approach starts with a comparison of concept names (labels) of two source schemata. If the comparison of concept names yields a match, the recognition approach continues with a comparison of the concepts' neighborhood. This means studying not only the direct connecting concepts (including their names), but also their dependencies for the concepts that yielded a match. In other words concept neighborhood is comprised of the concepts directly connected to the concept yielding a match.

![Diagram of recognition of homonyms based on both name and neighborhood matching](image)

**Figure 21. Recognition of homonyms based on both name and neighborhood matching**
The term dependency is used to represent the connection between two concepts for instance the (1, 1; 1, *) association dependency between Article and OrderItem in Schema 1. The proposed two step approach is illustrated in Figure 21.

The proposed dependency approach might also be expressed as follows. First the concept names of the two source schemata are compared. When the comparison yields a match, as in between Name in Schema 1 and Schema 2 in Figure 21, the dependency approach continues with comparison of the connected concepts including their dependencies (connections). In doing so, a (1, 1; 0, *) dependency is identified between Name and Customer in schema 1 and between Name and Product in schema 2. More precisely this means that there is no neighbor match since the connected concepts are named differently although the dependency, in this case association, and cardinality is the same (0, *; 1, 1). Having the same concept name in different contexts might indicate a homonym conflict. This means that when a name match has been recognized and the two concepts have different neighbors, are used in different context, a homonym conflict might have been recognized.

The proposed solution to the second problem (resolution) was to apply a resolution technique that combines prefixing (dot-notation “.”) and the inheritance dependency. Prefixing was introduced for compound concept names, to make it possible to unambiguously identify a concept in a schema (cf. Gustas 1997). The inheritance dependency was introduced since by applying prefixing it is possible to semantically enrich the integrated schema and prevent an impoverishment of the vocabulary and concept names used in the schemata. The proposed resolution technique is illustrated in Figure 22 where the homonyms have been resolved. In the merged schema the source schemata have been integrated over the Name concept. It should be noted that in Figure 22 some problems still exist. It should also be noted that while integrating two schemata the schema boundaries change as illustrated with the dotted line in Figure 22.
5.2.2 Inference Rules

In the second paper of this research topic (Bellström 2006b), I studied problems, challenges and opportunities surrounding the application of inference rules in schema integration. An example of how an inference rule might be applied to recognize and resolve inter-schema properties, including both inheritance and instance-of dependencies in schema integration, is illustrated in Figure 23, Figure 24 and Figure 25. The illustrated rule can be described as follows: If Sony_TV_32_001 is instance-of TV and TV inherits Product then Sony_TV_32_001 is instance-of Product. The primitives used in Figure 23, Figure 24 and Figure 25 are described in Figure 5. The given conditions between the source schemata are illustrated in Figure 23 and the product of applying the inference rule in Figure 24. Yet, it should be noted that the rule should only be applied to discuss the schemata. This is the case since in the integrated schema the TV concept should be, as illustrated in Figure 25, included otherwise we cause semantic loss, meaning one concept is lost.

![Figure 22. Integrated schema with resolved homonyms introducing a new concept and applying inheritance dependency together with sharper cardinality](Image)
To summarize, the first problem studied in Bellström (2006b) focused on *why* inference rules should be applied to recognize differences and similarities between two source schemata in schema integration. The second problem studied in the paper focused on *how* inference rules could be applied to recognize and resolve differences and similarities between two source schemata in schema integration. While studying both problems, the notation of the Enterprise Modeling approach was applied.

The proposed solution to the first problem (*why*) was to apply inference rules in the “if then” form (see Figure 23 and Figure 24) as a communication “tool” (reasoning rules) between users and/or designers.
The proposed solution to the second problem (how) was to apply inference rules to deduce new dependencies and concepts, where a difference or similarity has been identified between two compared source schemata. In Figure 26 and Figure 27 an example is given on how an inference rule could be applied as a communication “tool” and as a resolution technique for the synonyms recognized between Article and Product.

![Schema 1](image1.png)

![Schema 2](image2.png)

**Figure 26.** Example of inference rule applying the inheritance and the mutual inheritance dependencies

![Integrated Schema](image3.png)

**Figure 27.** Inference rule applying inheritance and mutual inheritance dependencies

The inference rule illustrated in Figure 27, used to resolve the synonyms recognized in Figure 26, could be described as follows: Product mutually inherits (↔) Article if and only if Product inherits (→) Article and Article inherits (→) Product. In other words, if Product inherits Article and Article inherits Product then we can truly say that Product and Article are synonyms.
5.2.3 Resolution of Homonyms, Synonyms and Hypernyms-Hyponyms

The third paper in this research topic (Bellström & Carlsson 2006) is also the first placed within research topic one and two, meaning it has contributions to both research topics. In the paper we studied the problems connected to resolution techniques for similarities, such as synonyms and the inter-schema hypernym-hyponym property and differences, such as homonyms, recognized between two source schemata. In the paper we focused on the vocabulary used in the conceptual database schemata, meaning the applied resolution techniques should resolve the problems without losing (deleting) any of the concepts including their names. In other words, the similarities and differences should be dealt with without impoverishing the vocabulary used in the conceptual database schemata. An example of the problem connected with the inter-schema hypernym-hyponym property, e.g. the use of concept name *Order* in Schema 1 and the use of concept name *High Priority Order* in Schema 2, is illustrated in Figure 28. The primitives used in Figure 28 are described in Figure 5.

![Figure 28. Example of concept names that might indicate an inter-schema hypernym-hyponym property](image)

To summarize, the problem studied in the paper was how to resolve synonyms, homonyms and inter-schema hypernym-hyponym properties without losing any of the used concepts including their names. This implies that the proposed resolution methods should work without impoverishing the vocabulary used in the conceptual database schemata. At the same time, the resolution methods should counter an occurrence of concept name compression and semantic loss. Emphasis was placed on the importance of using the same vocabulary while designing the conceptual database schemata and while verifying or declining the integrated conceptual database schema in schema integration. In the paper, the
notation given in the Enterprise Modeling approach was applied to illustrate the proposed resolution methods.

In the paper, “the line of mutual interpretation” notion was coined in connection with resolution methods for differences and similarities between two source schemata. The proposed solution to the problem of how to maximize the line of mutual interpretation was to apply resolution methods that did not lose (delete) any of the concepts, including their names. We illustrated and demonstrated how to retain the used concept names in the source schemata even after resolving homonyms, synonyms and inter-schema hypernym-hyponym properties. Retaining concept names should increase the possibility to mutually interpret the integrated schema and thereby increase and maximize the line of mutual interpretation. In Figure 29, the inter-schema hypernym-hyponym property recognized in Figure 28 has been resolved by applying the inheritance dependency between Order and High Priority Order. Otherwise all other concepts, without Priority connected to High Priority Order, have been connected to Order, the most general concept of Order and High Priority Order. It should be noted that an inter-schema property it is not really a conflict. Instead, an inter-schema property indicates how two concepts are related to each other by certain constraints (e.g. Batini et al. 1992; Johannesson 1993). The inter-schema hypernym-hyponym property constraints can be formulated as follows: an inter-schema hypernym-hyponym property occurs if one concept is defined as a hyponym (specialization) of another concept (cf. High Priority Order in Figure 29) and/or concept two is defined as a hypernym (generalization) of another concept (cf. Order in Figure 29).

![Figure 29. Integrated schema with resolved inter-schema hypernym-hyponym property](image-url)
In the paper, it was proposed that the introduction of the inheritance dependency between the two concepts resolves the inter-schema hypernym-hyponym property (cf. Order and High Priority Order in Figure 29). By introducing the inheritance dependency, the concepts and their names are retained after the inter-schema property has been resolved. Otherwise all concepts that are common are moved to the most generic concept.

**5.2.4 Recognition and Resolution of Similarities and Differences**

In the fourth paper of this research topic (Bellström et al. 2007), linguistic similarities and differences were studied. Focus was put on how to manually recognize and resolve linguistic similarities and differences in schema integration. More precisely, this means problems in connection with manual recognition and resolution of both differences (homonyms) and similarities (synonyms, hyponyms-hyperonyms and holonyms-meronyms) between two source schemata. Problems connected with the inter-schema holonym-meronym property, e.g. the use of concept name Order in Schema 1 and the use of concept name OrderItem in Schema 2, is illustrated in Figure 30. The proposed resolution technique for the recognized inter-schema property is illustrated in Figure 31. The primitives used in Figure 30 and Figure 31 are described in Figure 5.

**Figure 30.** Example of concept names and neighborhoods that might indicate an inter-schema holonym-meronym property
Figure 31. Integrated schema with resolved inter-schema holonym-meronym property

It should be noted that the dependency (connection/link) between Order and OrderItem are changed to composition. Composition is a dependency illustrating ‘part-of’ with strong ownership and coincidental lifetime.

To summarize, the first problem studied in the paper was how to manually recognize both linguistic differences and similarities between two conceptual database source schemata. The second problem studied was how to manually resolve linguistic differences and similarities recognized between two conceptual database source schemata. While studying both problems, the notation of the Enterprise Modeling approach was applied. The proposed solution to the problem of manual recognition of differences (homonyms) and similarities (synonyms and inter-schema properties) was to apply an extension of the dependency approach first described in Bellström (2005). This means that the names of two concepts from two source schemata are first compared and when the name comparison yields a match, the dependencies and concepts directly connected to the matched concepts are also compared. Figure 32 illustrates the proposed comparison approach where a match has been identified over the concept named OrderItem.
The following step is to compare the concept neighborhoods: comparing the connecting dependencies and concepts of the matched concepts, of OrderItem in both schema one and two. In doing so, a (1, 1; 1, *) dependency is identified between OrderItem and Article in Schema 1 and between OrderItem and Product in Schema 2. In addition, both Article and Product have a connected concept, via the (1, 1; 0, *) dependency. This information might indicate that two synonymous concept names (Article and Product) are used. In other words, when two concepts have similar neighbors, including both dependencies and concept names, but different names, synonyms, concept names might be recognized.

Concerning recognition of the inter-schema hypernym-hyponym property, it was proposed to carry out a similar procedure as described above: first name comparison followed by comparison of the neighborhood. However, when dealing with recognition of the inter-schema hypernym-hyponym property it should be noted that this type of inter-schema property might be treated as a synonym if not analyzed deeply enough since both problems are indicating similarity. The inter-schema hypernym-hyponym property instead describes a generalization – specialization dependency between two concepts as illustrated between Order and High Priority Order in Figure 29.

Finally, the proposed approach and solution to the problem of manual resolution of synonyms is to apply the mutual inheritance dependency (see Figure 26) and manual resolution of the hypernym-hyponym property is to apply the inheritance dependency (see Figure 29).
5.2.5 Semantic Loss in Relation to Recognition and Resolution of Similarities and Differences

The fifth and last paper in this research topic (Bellström 2009) is also the second paper which is placed within research topic one and two. In the paper, I studied problems connected with recognition and resolution of homonyms, synonyms, the hypernym-hyponym inter-schema property and the holonym-meronym inter-schema property. I also studied how minimality, the use of different modeling constructs and the use of non-comparable transformations influence the resolution of differences and similarities between two source schemata in schema integration. An example of the problem connected with the inter-schema holonym-meronym property and the inter-schema hypernym-hyponym property, e.g. the use of concept name Order and OrderItem in Schema 1 and the use of High Priority Order and OrderItem in Schema 2, is illustrated in Figure 33. The proposed resolution techniques for the recognized inter-schema properties are demonstrated and illustrated in Figure 29 and Figure 31. The primitives used in Figure 33 are described in Figure 5.

![Figure 33. Example of concept names that might indicate inter-schema properties](image)

To summarize, there are two main problems studied in the paper. The first problem concerns how minimality, the use of different modeling constructs and the use of non-comparable transformations influence resolution of differences and similarities between two source schemata in schema integration. The second problem concerns how to resolve differences, such as homonyms, and similarities, such as inter-schema properties, without losing any of the concepts and their names. In particular, focus was put on resolution methods that counter an impoverishment of the vocabulary used in the conceptual database schemata and at the same time prevent the occurrence of semantic loss.
In the last paper, methods for recognition and resolution of similarities and differences between two source schemata were studied. This means not only the recognition and resolution of homonyms, synonyms, but also, the recognition and resolution of hypernyms-hyponyms (including specialization, “subset-of” and “instance-of”) and holonyms-meronyms (including both aggregation and composition). In connection with the recognition of synonyms, the problem sometimes referred to as reverse subset relationships (Batini et al. 1992) and cyclic generalization (Song 1995) was also included. Reverse subset relationships and cyclic generalization occurs when a concept in schema 1 is defined as a hyponym and in schema 2 defined as a hypernym such as Article inherits Product (schema 1) and Product inherits Article (schema 2).

The problems mentioned were approached by first analyzing what minimality, the use of different modeling constructs and the use of non-comparable transformations actually mean for the vocabulary used in schemata in schema integration. At the same time, it was explored if minimality, the use of different modeling constructs and the use of non-comparable transformations might cause semantic loss. This was followed by a study of the most common resolution methods proposed in the literature. It was concluded that all the mentioned problems might cause semantic loss. In the paper, the proposed solution was to apply resolution techniques that do not lose (delete) concepts and/or dependencies but instead retain them. The core of the proposed resolution methods lies in the importance of using precise concept names instead of conducting renaming. A specific resolution technique highlighted in the paper was to the use of the instance-of dependency (Figure 35) to resolve both intra-schema and inter-schema problems (Figure 34).
The schemata shown in Figure 34 illustrate two ways of defining and describing a specific Type of Publication using specialization (Schema 1) and association (Schema 2). If the users, together with the designers, decide to adopt Schema 2 without documenting the actual Type(s) of Publication(s) illustrated with the inheritance dependency in Schema 1 (Report Publication, Conference Publication and Journal Publication) these concepts, including their names, are lost which causes semantic loss. On the other hand if the users and designers instead decide to adopt the schema illustrated in Schema 1 without the Publication.Type concept into consideration it might also cause semantic loss since the Publication.Type concept is lost which could be taken into consideration while adapting more precise concept names (labels) in the integrated conceptual database schema. However, if the users and designers decide to include the instance-of dependency (see Figure 35) all the concept names are retained.

![Figure 35. Applying instance-of dependency to resolve the specialization – association problem](image)

Retaining concept names (labels) counters an impoverishment of the vocabulary used in both the database source schemata and the global integrated conceptual schema. Users can then either verify or decline the integrated schema using their own concept names (labels). Studying the integrated schema in Figure 35 it can be concluded that schema 2 in Figure 34 actually hides an important dependency, where the instances of Publication.Type are actually subtypes of Publication. This is known as a power type. Martin and Odell (1998) explain the problem as follows:
A power type is a type whose instances are subtypes of another type. (p. 252).

As the definition given by Martin and Odell (1998) explains, a power type is a problem users and designers have to be aware of while studying one schema (intra-schema problem) and while comparing two source schemata (inter-schema problem) in schema integration. In the Unified Modeling Language Superstructure Specification, v2.2 (Object Management Group 2009) a power type is defined as:

 [...] a power type is a class whose instances are subclasses. (p. 55).

Finally, I want to strengthen the importance of applying the instance-of dependency. This is done quoting Hoppenbrouwers et al. (2005a) who discuss modeling scenarios as follows:

When discussing models with stakeholders and informants, in particular when trying to establish a common understanding, it is sensible to discuss different scenarios and alternatives to the model being considered. Doing so leads to an exploration of the meaning and impact of the model taking shape, and also leads to improved mutual understanding. (p. 271)

The schema in Figure 35 fulfills the importance of different scenarios and alternatives by including all concepts and introducing the instance-of dependency. Being able to illustrate both inheritance and instance-of dependencies in one schema is also a strength of applying the notation of the Enterprise Modeling approach.

5.3 Contribution

There are two main contributions given in the second research topic: Integration of Static Schemata. The first is given in the method to recognize differences and similarities between two source schemata.

The proposed method for the recognition of differences and similarities between two source schemata is an extension of the work presented in this research topic, meaning parts of it have been improved and several new parts have been added. The method starts with a comparison of concept names used in the compared conceptual database source schemata. When comparison of concept names yields either ‘equivalent’ or ‘similar’ the method continues with a comparison of the concept neighbors. In this thesis, neighbors, sometimes also
referred to as neighborhoods or surroundings, means the concepts and dependencies that are directly connected to the compared concepts. This is done by following the dependencies, connected to the matched concepts, to other concepts (see Bellström 2005; Bellström et al. 2007). The result of the two step comparison method might yield that either the compared concepts are equivalent, similar or different. Different means that the concepts are independent of each other. However, this distinction is seldom easy to distinguish. Therefore, the following “if- then” rules are proposed for equivalent concept names (Figure 36) and similar concept names (Figure 37). The arrows in Figure 36 and Figure 37 should only be viewed as one way to follow the alternative the method checks. For instance if the result of comparison of concept names yields equivalent and the result of comparison of concept neighborhoods yields equivalent then the method suggests that the compared concept names are equivalent (Rule E1).

Rule E1: If comparison of concept names (labels) yields equivalent, and comparison of concept neighborhoods yields equivalent then equivalent concepts might be recognized.

Rule E2: If comparison of concept names (labels) yields equivalent and comparison of concept neighborhoods yields different then homonyms might be recognized between the two concept names (labels) in the source schemata. For instance using concept name (label) Name in the context of Customer (schema one) and in the context of Product (schema two).

Rule E3: If comparison of concept names (labels) yields equivalent and comparison of concept neighborhoods yields similar, meaning one concept in each source schemata is named different, then synonyms might be recognized between the two concept names (labels). For instance using concept name (label) Article in schema one and Product in schema two.

Rule E4: If comparison of concept names (labels) yields equivalent and comparison of concept neighborhoods yields similar, meaning one concept name in one of the source schema is a composite of a concept name in the other source schema with a following addition, and cardinality indicates “one-to-one”, then an association dependency between the two concept names might be recognized. For instance using Customer in schema one and Customer Name in schema.
Rule E5: If comparison of concept names (labels) yields equivalent and comparison of concept neighborhoods yields similar, meaning one concept name in one of the source schemata is a composite of a concept name in the other source schema, with a prior addition, then a hypernym-hyponym dependency might be recognized between the two concept names (labels). For instance using High Priority Order (hyponym) in schema one and Order (hypernym) in schema two.

Rule E6: If comparison of concept names (labels) yields equivalent and comparison of concept neighborhoods yields similar, meaning one concept name in one of the source schema is a composite of a concept name in the other source schema with a following addition and cardinality indicates uniqueness with “many”, then a holonym-meronym dependency between the two concept names might be recognized. For instance using Order in schema one and Order Line in schema two.

Rule E7: If comparison of concept names (labels) yields equivalent and comparison of concept neighborhoods yields similar, meaning one concept name in one of the source schema is a composite of a concept name in the other source schema with a following addition and the cardinality between the two concepts indicates “many” without uniqueness, then a holonym-meronym dependency between the two concept names might be recognized. For instance using Order in schema one and Order Line in schema.

A few issues need to be addressed in the presented rules. Depending on how the dependencies are used in the source schemata rule four, five and six could indicate either sharper or looser dependencies. This means that rule E4 could indicate an association or a composition (part-of) dependency and rule E5 and E6 could indicate an association dependency.
Figure 36. Method for recognition of similarities and differences with focus on equivalent concept names (labels)
Rule S1: **If** comparison of concept names (labels) yields similar, meaning one concept name and one concept name with a following addition to the first one, **and** comparison of concept neighborhoods yields similar or equivalent with a “one-to-one” in cardinality **then** an association dependency between the two concept names might be recognized. For instance using Customer in schema one and Customer Name in schema.

Rule S2: **If** comparison of concept names (labels) yields similar, meaning one concept name and one concept name with a following addition to the first one, **and** comparison of concept neighborhoods yields similar or equivalent with uniqueness and “many” in cardinality **then** a holonym-meronym dependency between the two concept names might be recognized. For instance using Order in schema one and Order Line in schema two.

Rule S3: **If** comparison of concept names (labels) yields similar, meaning one concept name and one concept name with a following addition to the first one, **and** comparison of concept neighborhoods yields similar or equivalent and the cardinality indicates “many” without uniqueness **then** a holonym-meronym dependency between the two concept names might be recognized. For instance using Order in schema one and Order Line in schema two.

Rule S4: **If** comparison of concept names (labels) yields similar, meaning one concept name and one concept name with a prior addition to the first one, **and** comparison of concept neighborhoods yields similar or equivalent, **then** a hypernym-hyponym dependency might be recognized between the two concept names (labels). For instance using High Priority Order (hyponym) in schema one and Order (hypernym) in schema two.
Sometimes it could be useful to compare concept neighborhoods even if comparison of concept names (labels) yields different. In doing so, synonyms that otherwise could pass unnoticed may be recognized. The result of a comparison of concept neighborhoods, with the prior result of different from comparison of concept names, could result in three cases. If different then the concept names are different or if similar or equivalent then concept names are synonyms. In line with the issues addressed in connection with rules E4-E6 above rules S1-S3 could indicate either sharper or looser dependencies.

It should be noted that the rules should be viewed as a first step towards a semi-automatic method for recognition of similarities and differences between two source schemata. Following the classification of schema matching approaches given by Rahm & Bernstein (2001), the proposed method is classified as a composite schema-based matching approach. This is the case since the method starts with comparison of concept names (labels) followed by comparison of neighborhoods.

Finally, before moving on to the second contribution of research topic two, the term similar needs to be addressed in relation to comparison of concept neighborhoods. In this case, similar does not exclude other concepts and dependences in the source schemata. For instance, if a hypernym-hyponym
dependency has been recognized then both the hypernym, e.g., Order, and the hyponym, High Priority Order, might have additional concepts connected.

The second contribution is given in the proposed resolution methods for recognized differences and similarities between two source schemata. In the proposed resolution methods, the solution is to apply precise concept names and at the same time to retain all concepts names (see Table 17 and Table 18) used in the source schemata. In the proposed methods for inter-schema properties, the solution is to apply more precise dependencies (see Table 20 and Table 21).

As illustrated in Bellström (2005) by applying prefixing together with the inheritance dependency, homonyms can be resolved retaining all the used concept names. This means including the context of the concept into its name such as Product.Name and Customer.Name. Table 17 summarizes the most common resolution methods for homonyms given in the literature together with the proposed resolution method. The proposed resolution method for homonyms is also in line with how the problem should be resolved according to Martin and Odell (1998) who state that:

[...] we must support the homonym phenomenon in ways that can be understood by humans–through context. (p. 20).

<table>
<thead>
<tr>
<th>PROPOSED TECHNIQUES</th>
<th>RESOLUTION TECHNIQUES</th>
<th>REFERENCES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Renaming one or two names</td>
<td>Lee &amp; Ling (2003), Mannino (2007), Spaccapietra &amp; Parent (1994)</td>
<td></td>
</tr>
<tr>
<td>Prefixing the names</td>
<td>Engels et al. (1992), Storey (1993)</td>
<td></td>
</tr>
<tr>
<td>Standardization of names</td>
<td>Lawrence &amp; Barker (2001), Maier et al. (1984)</td>
<td></td>
</tr>
<tr>
<td>Introducing prefixing (dot-notation) together with the inheritance dependency.</td>
<td>Bellström (2005, 2006b)</td>
<td></td>
</tr>
</tbody>
</table>

As illustrated in Bellström & Carlsson (2006), by applying the mutual inheritance dependency to resolve synonyms, concept names are retained and the users are able to use their own vocabulary while verifying or declining the integrated schema. Table 18 summarizes the most common resolution methods for synonyms given in the literature together with the proposed resolution method. This is also in line with the following statement given by Martin and Odell (1998):
While standardization is useful, enforcing the one-and-only-one name is impractical in all situations. (p. 19).

Finally, before discussing resolution methods for similar concept names, it is important to once more stress the importance of resolving homonyms and synonyms in the schema integration process. This is done by quoting Mannino (2007) who express it as follows:

In any integration approach, resolution of synonyms and homonyms is a very important issue. (p. 441).

<table>
<thead>
<tr>
<th>PROPOSED RESOLUTION TECHNIQUES</th>
<th>REFERENCES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Renaming one or two names</td>
<td>Lee &amp; Ling (2003), Spacecpietra &amp; Parent (1994)</td>
</tr>
<tr>
<td>Rename or establish a list of synonyms</td>
<td>Mannino (2007)</td>
</tr>
<tr>
<td>Provide aliases</td>
<td>Parent &amp; Spacecpietra (1998)</td>
</tr>
<tr>
<td>Introduce the mutual inheritance dependency between the two concepts.</td>
<td>Bellström (2006b), Bellström &amp; Carlsson (2006)</td>
</tr>
</tbody>
</table>

As illustrated in Bellström (2009), reverse subset relationships and cyclic generalization might instead indicate synonyms. If so, it should be resolved by applying the mutual inheritance dependency. However, if it is a true reverse subset relationships or cyclic generalization, concepts should be given more precise names without losing any of the original ones. In Table 19 the most common resolution methods for cyclic generalization and reverse subset relationships are given together with the proposed resolution method.
Table 19. Resolution methods for cyclic generalization and reverse subset relationships

<table>
<thead>
<tr>
<th>PROPOSED RESOLUTION TECHNIQUES</th>
<th>REFERENCES</th>
</tr>
</thead>
<tbody>
<tr>
<td>“[...] re-consider more precise concepts or rename some words [...].”</td>
<td>Song (1995 p. 110)</td>
</tr>
<tr>
<td>“[...] selecting one representation over the other or building a common representation [...]”</td>
<td>Batini et al. (1992 p. 128)</td>
</tr>
<tr>
<td>If it is a true cyclic generalization or reverse subset relationships apply more precise concept names. However, if it instead indicates synonyms introduce the mutual inheritance relationship between the two concepts.</td>
<td>Bellström (2009), Bellström &amp; Carlsson (2006)</td>
</tr>
</tbody>
</table>

Inter-schema properties express similarity between two concepts and are also resolved introducing a certain dependency. More precisely, this means that if an inter-schema hypernym-hyponym property has been identified between two concepts it could be resolved introducing either the inheritance dependency or the subset-of dependency between the two concepts. Table 20 summarizes the most common resolution methods for the inter-schema hypernym-hyponym property given in the literature together with the proposed resolution method.

Table 20. Resolution methods for the inter-schema hypernym-hyponym property

<table>
<thead>
<tr>
<th>PROPOSED RESOLUTION TECHNIQUES</th>
<th>REFERENCES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduce an is-a relationship</td>
<td>Lee &amp; Ling (2003), Storey (1993)</td>
</tr>
<tr>
<td>When concepts are not exclusive, apply the inheritance dependency between the two concepts.</td>
<td>Bellström (2009), Bellström et al. (2007)</td>
</tr>
<tr>
<td>When the concepts are exclusive, apply the subset-of dependency between the concepts.</td>
<td>Bellström (2009)</td>
</tr>
</tbody>
</table>

If an inter-schema holonym-meronym property has been identified between two concepts it could be resolved introducing either the aggregation or the composition dependency between the two concepts. Table 21 summarizes the most common resolution methods for inter-schema holonym-meronym property given in the literature together with the proposed resolution method.
Table 21. Resolution methods for the inter-schema holonym-meronym property

<table>
<thead>
<tr>
<th>PROPOSED RESOLUTION TECHNIQUES</th>
<th>REFERENCES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduce a part-of relationship</td>
<td>Storey (1993, 2001)</td>
</tr>
<tr>
<td>When the cardinality between the concepts recognized as an inter-schema holonym-meronym property contains a zero apply the aggregation dependency between the two concepts.</td>
<td>Bellström (2009), Bellström et al. (2007)</td>
</tr>
<tr>
<td>When the cardinality between the concepts recognized as an inter-schema holonym-meronym property does not contains a zero apply the composition dependency between the two concepts.</td>
<td>Bellström (2009), Bellström et al. (2007)</td>
</tr>
</tbody>
</table>

Finally, by applying the proposed resolution methods, all concept names are retained. This contributes to a richer schema in the sense that the users might use their own vocabulary while verifying or declining the integrated schema. The proposed resolution methods also counter the occurrence of concept name compression and semantic loss.

5.4 Contribution and Evaluation in Relation to Design Science

In design science, the research contribution is given in the IT-artifact. An artifact is a construct, a model, a method or an instantiation (Hevner et al. 2004; March & Smith 1995). The first contribution of the second research topic is classified as a method. The method could also be viewed as an algorithm. March & Smith (1995) describe a method in design science research as follows:

A method is a set of steps (an algorithm or guideline) used to perform a task (p. 257).

The task to perform in this case is the recognition of similarities and differences between two source schemata. The method is useful since in applying it both similarities and differences that otherwise might pass unnoticed can be recognized. In the recognition method, a combination of concept name comparison and concept neighborhoods comparison is applied, a so-called composite schema-based matching approach (Rahm & Bernstein 2001). The method is original since it is applied on schemata that are designed on an implementation independent level focusing on concepts and their names, and dependencies (connections/links) between them.
The recognition method was evaluated applying ‘informed arguments’ and ‘scenarios’ which are both descriptive evaluation methods (Hevner et al. 2004). Informed arguments were used since both weaknesses and strengths of relevant research put the method into context and demonstrate its utility. On the other hand, scenarios were used not only to demonstrate the utility of the method, the artifact, but also to apply, test and compare it with earlier proposed recognition methods and already existing integration examples.

Although the method might be viewed as an algorithm, I do not claim having implemented it in a prototype application. If that had been the case, the contribution would have instead been classified as an instantiation and the evaluation method could instead have been testing. Nevertheless, creating a prototype application, an instantiation, is relevant for future research.

The second contribution of research topic two is also classified as a method. In this case the method, the artifact, is instead a resolution of similarities and differences that have been recognized between two source schemata. The method is original since it is applied on similarities and differences recognized between two source schemata that are described on an implementation independent level. At the same time, this means that focus should be on concepts and their names, and dependencies (connections/links) between these instead of minimizing the number of concepts and dependencies in the integrated schema. For instance, in the resolution method for homonyms (see Table 17) a new concept including the context of the concept name is included, which inherits the more generic concept which had been recognized as a homonym. This resolution method shows that all concept names are retained even after resolving the problem. On the other hand, it should be noted that if the user or designer is instead taking an implementation dependent perspective he or she could argue that the schema is slightly over-specified and that the new concept name should be applied.

The resolution method was validated using two descriptive evolution methods: ‘informed arguments’ and ‘scenarios’. These two evaluation methods are described as follows:

Informed Argument: Use information from the knowledge base (e.g., relevant research) to build a convincing argument for the artifact's utility […]
Scenarios: Construct detailed scenarios around the artifact to demonstrate its utility (Hevner et al. 2004 p. 86).

Although, it could also be argued, similar to validating the second contribution of research topic one, that one analytic evaluation method also has been applied: 'static analysis':

Static Analysis: Examine structure of artifact for static qualities (e.g., complexity) […] (Hevner et al. 2004 p. 86).

In this case 'static analysis’ was applied to evaluate the ability, quality (Hevner et al. 2004), to retain all concept names, used in the source schemata, in the integrated schema.
6 Integration of Static and Dynamic Schemata

The purpose of this chapter is to present a summary of the third research topic. This is achieved by addressing the motivation, problem, approach and solution and finally contribution to the field. In the first section the research topic is introduced and motivated. In the second section the solution to the addressed problems is presented. Finally, in the third section the contributions found in this research topic are presented.

6.1 Motivation

While designing databases and information systems it is important to carry out conceptual design of both the static and the dynamic aspects. The conceptual database schemata illustrate the static aspects of the future information system. This means not only what the future information system needs to store in the database but also what data is needed by the information system during processing. On the other hand, the dynamic schemata illustrate the dynamic aspects of the future information system, meaning what functionality the future information system should provide. While carrying out conceptual design, it is important to integrate all schemata into one schema since one view of the future information system is needed for both further design and development. However, schema integration is a difficult and time consuming task and methods and modeling approaches are needed to facilitate the integration process. Batini et al. (1992) describe schemata integration in relation to integration of static schemata as:

\[\ldots\] the process of merging several conceptual schemas into a global conceptual scheme that represents all the requirements of the application. (p. 119).

The first problem studied in this research topic deals with how a generic method for integrating not only static schemata and dynamic schemata but also the static and the dynamic schemata into one schema should be defined and designed.

It should be noted that in the mid 1980’s integration of dynamic schemata was already mentioned as a topic where research was required (see for instance Batini et al. 1986). However, the research topic has until recently been viewed as very difficult and therefore often neglected in favor of integration of static schemata. For instance in Conrad et al. (2002), the authors stated that

improved support for integrating behaviour (p. 250)
is an open question and therefore also a topic for future work. In Stumptner et al. (2004), the authors go even one step further and describe the importance of integration of dynamic schemata as follows:

The goal for the upcoming years clearly is to develop methods and techniques for the logical integration of object behaviour which may well extent to the integration of whole object-life cycles. (p. 15)

The second problem studied in this research topic deals with states and state changes in the database and how to illustrate these in the dynamic schemata. In doing so we are able to extend conceptual database design with states and state changes. Included in this last problem is how to analyze and design active database rules (Ceri & Ramakrishnan 1996; Paton & Díaz 1999; Widom & Ceri 1996) on a conceptual level and at the same time include states and state changes into these activities.

In both research problems, the notation given in the Enterprise Modeling approach (Gustas 1998; Gustas & Gustiené 2004) has been chosen as the modeling language. Because by applying that notation it is possible to not only describe and illustrate the static and the dynamic aspects of a future information system but also to integrate these aspects into one schema. Another motivation for using that notation is that its foundations are a modeling language making it possible to illustrate the future information system on both a conceptual and a logical level.

From that short introduction and motivation the papers referred to as Bellström & Jakobsson (2006, 2008), Jakobsson & Bellström (2007), Bellström et al. (2008, 2009) are placed within the third research topic: Integration of Static and Dynamic Schemata.

6.2 Problem, Approach and Solution

One of the main problems studied in this research topic focuses on how to develop a method in which both static and dynamic schemata are designed and integrated into one description of the future database and information system. A second problem addressed in this research topic focuses on how to develop a method making it possible to design and integrate legal states and state changes in a database and to illustrate these in the conceptual database schema. This means that this research topic deals with problems connected with design and
integration of schemata including not only static and dynamic aspects but also conceptual and logical contents (syntactic elements). It also means that the important topics of database consistency and integrity are dealt with.

6.2.1 A Generic Method to Designing and Integrating Static and Dynamic Database Schemata

In the first paper of this research topic (Bellström & Jakobsson 2006) the problems concerning how to design and integrate not only static and dynamic schemata but also how to integrate these into one schema were studied. In doing so we addressed the question of what a generic and integrated method (in the paper referred to as an approach) to designing static and dynamic schemata should look like and how it could empower the design process including both design and integration. In the paper we studied not only how to recognize and resolve similarities and differences between two source schemata but also important issues such as having one global schema describing the information system. In this context, a global schema means having one integrated schema comprised of both static and dynamic aspects.

The problem was studied applying the notation given in the Enterprise Modeling approach. This approach has an advantage, since in applying it, it is possible to first design the static schemata and then design the dynamic schemata, and at the same time include parts of the designed static schemata. In the step that follows after schema design it is possible to integrate the designed schemata into one schema. This means first that we can concentrate on the static aspects and integrate the schemata containing these. After that we can concentrate on the dynamic aspects and integrate the schemata containing both the static and the dynamic aspects into one schema. The method (proposed in the paper) is illustrated in Figure 38.
Figure 38 should be interpreted as follows: input to view modeling is customer requirements (Customer Requirements (1) and Customer Requirements (2) in the top of Figure 38). With that input, database view modeling is conducted resulting in a set of schemata illustrating the static aspects (Database View Modeling). After that both the customer requirements and the static schemata are used while conducting software component view modeling resulting in a set of schemata illustrating the dynamic aspects (Software Component View Modeling). This is followed by integrating the static schemata into one global conceptual database schema (Conceptual Database Schema). Finally, the dynamic schemata are integrated into one global dynamic schema (Conceptual Software Components Schema). Input to this step is both the integrated static schema and the set of designed schemata illustrating the dynamic aspects. Finally, before delivering the global integrated schema, including both the static and the dynamic aspects, it is checked that the global schema is consistent. It should be noted that some of the generic guidelines proposed in Bellström et al. (2008) could be applied to facilitate integration.
While designing the schemata of the future database and information system, users and designers are focusing on different parts which often overlap. In the end, these schemata should be integrated into one schema, since it is one database that is being developed. We also need to recognize and resolve differences and similarities between two source schemata before moving on with further design and development. Let us therefore, as an illustrating example, consider the two schemata in Figure 39 and Figure 40. In Figure 39 the users are from the sales department focusing on selling goods while the users in Figure 40 are from the warehouse department focusing on printing an order of the ordered goods. Prior to designing the dynamic schemata, illustrated in Figure 39 and Figure 40, the static schemata for each department has already been designed. The primitives used Figure 39 and Figure 40 are described in Figure 5, Figure 6 and Figure 7.

In Figure 39 a User is creating an Order by the Create Order action. The data about an Order is typed into the SalesSC1 screen and thereafter sent to the Sales Component. The Sales Component then stores the data about an Order to the database DB by the Store Order action. It should be noted that after the Create Order action the state of Order is Order [Created] and after the Store Order action the state of Order is Order [Stored].

![Figure 39. Schema 1: Sales department (adapted and modified from Bellström & Jakobsson (2006 p. 640))](image-url)
On the other hand, in Figure 40 the Sales Component sends an Id to the Warehouse Component through the Get Order action. The Warehouse Component then uses the Id to collect the stored Order. This is done by the Return Order action resulting in the state of Order [Returned]. Finally, the Warehouse Component prints the order through the Print Order action and in the end the User receives a printed Order Sheet. The following step in the method is integrating the static schemata. Figure 41 illustrates the result of that step.

In the step that follows integration of static schemata, the dynamic schemata are integrated based on both the integrated static schema and the two dynamic schemata illustrating the user views for sales and warehouse department. It
should be noted that some static concepts have been given more precise names, such as Order.Id and Customer.Name, in the integrated static schema and it is these names that are used while integrating the dynamic schemata. It should also be noted that while integrating two schemata the system boundary changes and expands since parts are integrated into one. Let us now take a closer look at the integration procedure and its result. The overlap between the schemata illustrating the sales and warehouse department is found in the Sales Component. Sales Component is present in both schemata using the Store Order (Schema 1 Figure 39) and the Get Order (Schema 2 Figure 40) action. Besides that both schemata are processing and storing data about orders which emphasize integration even more. The schemata are therefore merged over the Sales Component. On the other hand, to end up with a correct global schema the following had to be changed and added: more precise names not only for static concepts, e.g. Order.Id, Customer.Id and OrderLine.Quantity, but also for states, e.g. Order.Id [Created at Warehouse Component], and actions, e.g. Get Order.Id and Collect Order.Id were introduced. Besides that several additional dependencies were introduced. For instance an inheritance dependency between Order [Returned] and Order [Stored] and an association dependency between the new introduced state Order.Id [Collected] and Order [Returned]. Finally, some concepts were removed from state Order [Returned]. Because the inheritance dependency between Order [Returned] and Order [Stored] was introduced making it possible to access these concepts through that link. The global integrated schema is illustrated in Figure 42.
6.2.2 Modeling and Integrating States and State Changes to Ensure Database Consistency

In the second paper of this research topic (Jakobsson & Bellström 2007) we studied how to, on a conceptual level including some syntactic elements, describe (model) legal states and state changes in the future database. More precisely, this meant describing how to model and integrate legal states and state changes in a database and to illustrate state and state changes as pre- and post-conditions in the conceptual database schema.

By doing so, we emphasized the importance of applying a modeling language suitable for that purpose. We addressed the problem on a higher level of abstraction compared to more traditional modeling languages such as the Unified Modeling Language. We therefore chose the notation given in the Enterprise Modeling approach, since by applying the Enterprise Modeling
approach it is possible to integrate both static and dynamic schemata into one global integrated schema.

To end up with one schema, the problem of inconsistency was studied from two perspectives. Firstly, we looked at how to avoid inconsistency, such as differences and similarities, between two conceptual database schemata. Differences and similarities are often referred to as conflicts in schema integration (Lee & Ling 2003; Parent & Spaccapietra 1998). Secondly, we studied how to avoid inconsistency in the database. More precisely this meant how to illustrate legal states and state changes in the database. In database literature, states and state changes are referred to as database transaction (Connolly & Begg 2010; Haerder & Reuter 1983). While studying states and state changes, responsibilities between different actors, such as a database or a software component, were also included.

Figure 43 illustrates a state change from *State S* to *State U* including an intermediate state, *State C*, in the *Order* concept (class). The schema illustrated in Figure 43 is incomplete since actors and communication flows are not shown. The primitives used in Figure 43 are described in Figure 5 and Figure 6.

![Figure 43. Change of database state illustrated as pre- and post-conditions (adapted and modified from Jakobsson & Bellström (2007 p. 42))](image)

The solid black rectangle in Figure 43 highlights that a change of state in the database is actually composed of several actions and states in both the database and the information system including the intermediate software component state (named *State C* in Figure 43). This means that the responsibility of database consistency is shared between two actors: the database and the software component.
To demonstrate and illustrate the proposed solution, we gave an example system for creating, storing and updating an order. Figure 44 illustrates the already integrated static schemata. Figure 45 illustrates the creation and storing of an order while Figure 46 shows the changing and updating of an order.

![Diagram](image)

**Figure 44.** Integrated static schema (adapted and modified from Jakobsson & Bellström (2007, p. 40))

In the demonstrated and illustrated example system we focused on states in connection with a successful, e.g. Result [OK], versus an unsuccessful, e.g. Result [Not OK], database action.

In Figure 45 the User is creating and storing an Order. The User is doing this by communicating with the system through screen layout Sales SC1, Re-Enter Order and Order Stored. The post-condition and state after the Create Order action is Order [Created] synonymous with Order [Not Stored] an intermediate state in the Sales Component. In contrast, the Order [Created] state is a pre-condition for action Store Order which results in state Order [Stored] which is a persistent state in the database DB. Finally, the state Order [Created] is treated as synonymous with Order [Not Stored] while Order [Stored] is associated with Result [OK] and Order [Not Stored] with Result [Not OK].
Meanwhile, in Figure 46 the User is changing and updating an already existing order. The User is doing this by communicating with the system through screen layout Sales SC2, Re-Enter Order and Order Updated. The state and post-condition Order [Stored] is very important while integrating the schemata illustrated in Figure 45 and Figure 46. The motivation for this is that before a User can change and later update an Order in the DB, that specific Order has to be already stored in the DB. When the Order is stored and the DB has the persistent state Order [Stored] then, and only then, can the User Change the Order resulting in the intermediate state Order [Changed] synonymous with Order [Not updated] which at the same time is a pre-condition for action Update Order. This means that before updating an Order, the Order has to be changed. Finally, in Figure 46 state Order [Changed] and Order [Not Updated] are treated as synonyms while Order [Updated] is associated with Result [OK] and Order [Not Updated] with Result [Not OK].
Figure 46. Schema 2: Update order (adapted and modified from Jakobsson & Bellström (2007 p. 43))

It should be noted that the sequence of actions and states, both persistent and intermediate states, in Figure 46 are the same as the ones illustrated in Figure 43 in which database consistency is highlighted. Let us now take a quick look at the integrated schema illustrated in Figure 47.

The overlap between the schemata illustrated in Figure 45 and Figure 46 is found in the processing of an Order. However, the concrete links between the schemata is given between state Order [Stored] and Order [Updated] and between Order [Changed] and Order [Not Stored]. More precisely this means that Order [Updated] inherits Order [Stored] and Order [Changed] inherits Order [Not Stored]. Inheritance dependencies are therefore introduced and placed between the states. In addition two rather large changes are introduced. In Figure 45 Order [Created] and Order [Not Stored] were marked as synonyms which was also the case between Order [Changed] and Order [Not Updated] in Figure 46. Both these dependencies are changed to inheritance. The motivation for this is that in the integrated schema it is shown that these are not truly synonyms but are instead dependent through the inheritance dependency.
6.2.3 Modeling and Integrating Active Database Rules to Ensure Database Consistency

In the third paper of this research topic (Bellström & Jakobsson 2008), we studied how to design, illustrate and integrate active database rules in conceptual database design. In the literature, active database rules are often referred to as Event-Condition-Actions rules or ECA rules (Goldin et al. 2004; Paton & Dias 1999; Widom & Ceri 1996). While designing and illustrating an
active database rule we applied the notation given in the Enterprise Modeling approach.

The approach to the problem is a bit different compared to the other research problems dealt with in this research topic. Instead we started out by implementing a database and two applications. This was followed by a reverse engineering project on the implemented database and applications. More precisely this meant creating a MySQL (www.mysql.com) database. Creating the database included the creation of the database as such, the creation of tables, implementing a stored procedure and finally implementing a trigger. This was followed by implementing two applications: one web based PHP (www.php.net) application and one stand alone Java (java.sun.com) application. In both these applications, calls were made to the stored procedure which included a constraint, and from that it was decided to go ahead and update or not. If the decision was to go ahead and update then the trigger was also invoked. If the trigger was invoked it was checked whether the ordered amount was increased and if so, the amount stored in product had to be decreased. On the other hand, if the ordered amount instead was decreased the amount in product had to be increased. A modified schema of the described active database rule and the described constraint is illustrated in Figure 48. The motivation for the modification was that we conducted reverse engineering resulting in a schema close to implementation and in this chapter I want to illustrate the active database rule on a conceptual level. Finally it should be noted that in the paper, focus was placed on what Pation & Dias (1999) called the knowledge model which

\[ \ldots \] indicates what can be said about active rules in that system. (p. 67)

The most important part in Figure 48 is marked with gray background starting with the constraint stating that \( \text{OrderLine} \; [\text{Product.Quantity} \; > \; \text{OrderLine.Data.Quantity} \; - \; \text{OrderLine.Quantity}] \). The constraint is stating that the actual stored amount, of the specific Product that is changed in the OrderLine, has to exceed the incoming new OrderLine.Data.Quantity minus the old and currently stored OrderLine.Quantity. In other words the constraint is making sure that it is only possible to increase OrderLine.Quantity if the stock of the specified Product is large enough. This is followed by the Update action that is resulting in, if successfully updated, the state OrderLine [Updated]. However, it should be noted that between the described constraint and the conditions Product [OrderLine
Data.Quantity < OrderLine.Quantity] and Product [OrderLine Data.Quantity > OrderLine.Quantity] a composition dependency exists. This means that the described constraint and conditions are synchronized. In other words either both are successfully executed or none of them are. When the Update action is executed successfully both the OrderLine and the Product are updated. This means that when the OrderLine is updated it has to be decided whether to Increase or Decrease Product.Quantity. This results in Product [Updated] with the associated concept Updated Product.Quantity. Between OrderLine [Updated] and Product [Updated] a composition dependency also exists. This illustrates synchronization between these states.

Finally, if the Update action is executed then the Event-Condition-Action rule is also triggered and executed since the Update action corresponds to an event. In Figure 48 the conditions stating Product [OrderLine Data.Quantity < OrderLine.Quantity] and Product [OrderLine Data.Quantity > OrderLine.Quantity] are corresponding to the condition part in the ECA-rule while the Increase and Decrease actions are corresponding to the action part in the ECA-rule.
6.2.4 Generic Guidelines for Integration of Dynamic Schemata

In both the fourth and fifth paper of this research topic (Bellström et al. 2008, 2009) we embraced a slightly different focus and approach. More precisely, this meant only focusing on actions and pre- and post-conditions. In both papers, actions and conditions were viewed as the minimal constituents needed for modeling dynamic schemata. The consequence of only using conditions together with actions is that the proposed integration techniques focus on linguistic instruments rather than modeling constitutes. Another difference is
that the contribution in these two papers should be viewed as a first step towards a computer aided semi-automatic schema integration process.

In the fourth paper of this research topic (Bellström et al. 2008) we studied how to integrate dynamic schemata including only actions and pre- and post-conditions. More precisely this meant answering the question concerning which approaches have in the past been proved to not only be useful, but also successful while conducting integration of dynamic schemata. Even though the research reported in the paper should be viewed as a first step towards a computer aided semi-automatic integration approach, users are still very important during the whole schema integration process, since they possess the domain knowledge needed to either verify or reject the integrated dynamic schema made up of actions and conditions.

To summarize, the main problem studied in Bellström et al. (2008) concerns what approaches have been proved to be both useful and successful, while dealing with integration of dynamic schemata. It is important to point out that the approach should be generic enough to be applicable for modeling languages only made up of actions and conditions.

The problem was addressed with the assumption that a dynamic schemata might be modeled using only actions and pre- and post-conditions. This means that dynamic schemata illustrate and contain a sequence of concepts with pre-conditions, e.g. Book [Not Reserved], followed by an action, e.g. Reserve Book, and ending with a concept with post-conditions, e.g. Book [Reserved] (Figure 49). It should be noted that a post-condition of one action also might be a pre-condition of a following action and that a condition might be viewed as a state of a concept (object). It should also be noted that both condition Book [Not Reserved] and Book [Reserved] match two persistent states in the database. Since in the database a Book is either stored as Book [Not Reserved] or as Book [Reserved] contributing to database consistency.
As a result of the problem studied and addressed in Bellström et al. (2008), six generic guidelines useful for integration of dynamic schemata were proposed. Since dynamic and static schemata are closely connected and similar problems might occur, most of the proposed guidelines described below are also applicable in integration of static schemata. Nevertheless, guideline five, pattern based resolution of integration conflicts, is only applicable while integrating dynamic schemata. It should also be noted that some of the guidelines were applied integrating the schemata in Figure 39 and Figure 40 and Figure 45 Figure 46. The proposed guidelines are:

1. **Performing schema integration on the pre-design level**
   
   Schema integration should be performed very early in the information systems development process. This means that integration is done on a high level of abstraction. Detailed design and implementation issues are therefore postponed which also results in fewer conflicts.

2. **Standardizing concept notions and utilizing them during integration**

   Concept notions are standardized, e.g. plurals “Books” becomes singular “Book”, in a way that facilitates the integration process not only for users, but also for a computerized integration tool.

3. **Using domain repositories for supporting the integration process**

   Repositories such as ontologies, taxonomies and lexicons are proposed to facilitate the integration process. Repositories are proposed since they store not only concept notions, but also semantic relationships between concepts that are useful while comparing concept names trying to recognize differences and similarities between two source schemata.
4. **Neighborhood based conflict recognition**
   Not only concept names but also concepts’ neighborhood (structure) should be compared while comparing two source schemata, trying to recognize differences and similarities between them. While comparing concept neighborhoods both differences and similarities between two source schemata might be identified that otherwise could pass unnoticed.

5. **Pattern based resolution of integration conflicts**
   Relationship patterns such as parallel, disjoint, consecutive, alternative and mixed as described by Frank & Eder (1998) should be applied as a first classification of schema relationships. Based on these patterns the relationship types can be identified and fitting integration proposals can be proposed.

6. **Computer supported integration with utilizing user feedback**
   Computer based applications should be used to support not only the recognition of similarities and differences between two source schemata but also to support how to solve the recognized differences and similarities between two source schemata. However, it is strongly recommended to involve users to ensure a correct integration, since full automation is not possible due to its complexity.

### 6.2.5 A Method for Integrating Conditions (States) and Actions

In the fifth and last paper of this research topic (Bellström et al. 2009) we studied how a method for integration of dynamic schemata comprised of states and actions could be described and designed. In the paper, we also studied how the generic guidelines proposed in Bellström et al. (2008) could be applied. More precisely this means to which sub-phase(s) could and should each of the proposed guidelines be applied to facilitate schema integration. In the paper, it was again presumed that the source schemata might be modeled using only conditions (pre and post) and actions. For that reason the given integration method should be conducted as early as possible during systems development.

To summarize, the two problems studied in Bellström et al. (2009) were: what phases are needed for integration of dynamic schemata made up of actions and
conditions and in what phase(s) should the guidelines proposed in Bellström et al. (2008) be applied?

As a result, we proposed a method comprised of four phases for integration of dynamic schemata made up of actions and conditions. The first phase was named Preparation of Source Schemata. In it at least guideline 1 Performing schema integration on the pre-design level, and guideline 2 Standardizing concept notions and utilizing them during integration are relevant and should therefore be applied. Guideline 1 means that detailed design and implementation issues are not dealt with, resulting in fewer structural conflicts. The schemata should also be less complex and therefore easier to interpret for users, making external involvement a natural part of the integration method. The main goal in connection with guideline 2 was to ensure that meta-information about a concept is made explicit via the concept name. This should make the schemata easily interpretable for both users and a computerized application. Finally, in the first phase, recognition and resolution of intra-schema similarities and differences, such as synonyms and homonyms, were dealt with by applying guidelines 3-6 to a single source schema.

The second phase in the integration method was named Recognition of Conflicts and Commonalities between Source Schemata. In this phase guidelines 3 and 4 were suggested as relevant guidelines. However, it should be pointed out that guideline 6 is also relevant in this phase. The motivation for applying guideline 6 in phase 2 is that computer applications supporting and facilitating the integration method are needed. On the other hand, users should only need to verify or reject the proposals once: when all similarities and differences between two source schemata have been recognized and reported by the computer applications. Domain repositories (guideline 3) are used not only for facilitating computer supported schema matching, but also for concept disambiguation. Finally, by comparing the concepts’ neighborhood (guideline 4), i.e. their structure, similarities and differences such as synonyms and homonyms between two source schemata that otherwise might pass unnoticed might be recognized.

Figure 50 and Figure 51 illustrate two source schemata before (a) and after (b) applying guidelines 1-3. The schemata in the left part of Figure 50 and Figure 51 are transformed to the schemata in the right part of Figure 50 and Figure 51 using only actions and conditions together with explaining text. The notation in
the right part of Figure 50 and Figure 51 should be interpreted as follows: actors are marked with bold and italic text, User and Reservation System, static concepts are marked with bold text, Book, and flows with underlined text Customer & Book Data.

Finally, if a concept state (condition) e.g. Book [Reserved] has another concept associated with it and the static schemata is not created yet, it is very important to construct a static schemata while applying guidelines 1-3. Otherwise it might cause semantic loss since one or several concepts might be deleted and/or lost during the schema transformation. In the example schemata illustrated in Figure 50 and Figure 51, this should result in a static schema focusing on the Book concept. A Book would then at least have four connected concepts: Title, Author, ISBN and Status. Finally, the schemata would be merged over the state (condition) Book [Reserved] (see Figure 52).
In Figure 50 the Book [Reserved] condition (state) is a post-condition after the action in which a book is reserved and in Figure 51 a pre-condition for the action of cancel a book reservation.

The third phase in the integration method was named Resolution of Conflicts and Commonalities between Source Schemata. In this phase guideline 5 Pattern based resolution of integration conflicts and guideline 6 Computer supported integration with utilizing user feedback were mentioned as relevant. Guideline 5 means that after possible similarities and differences have been recognized in phase 2, the relationship types described in Frank and Eder (1998) can be used as a first method for classifying schema relationship patterns. Although the proposed integration method aims to automate the integration process as far as possible, full automation is not realistic due to its complexity (Badia 2004; Stumptner et al. 2004). Therefore user verification is very important not only to verify a correct integrated schema, but also to reject an incorrect integrated schema. Guideline 6 corresponds to these two issues.

Finally, the fourth phase was named Merging the Source Schemata and Restructuring the Global Schema. In this phase guideline 3 Using domain repositories for supporting the integration process and guideline 6 Computer supported integration with utilizing user feedback are relevant. In the paper this is not explicitly mentioned in the text. However, since inter-schema properties might be recognized between two source schemata that are to be merged, domain repositories can be used to retrieve the dependency to include between two concepts. Computer support
with user feedback characterizes the whole proposed integration process and should therefore also be marked as an important guideline to be used in the fourth and last phase. Table 22 summarizes the described phases of the proposed integration method. Table 22 also illustrates which guidelines (see Bellström et al. 2008) are useful and applicable in each of the phases in the proposed integration method.

Table 22. Guidelines and integration method

<table>
<thead>
<tr>
<th>PHASE</th>
<th>GUIDELINE</th>
<th>1. PERFORMING SCHEMA INTEGRATION ON THE PRE-DESIGN LEVEL</th>
<th>2. STANDARDIZING CONCEPT NOTIONS DURING INTEGRATION</th>
<th>3. USING DOMAIN REPOSITORIES FOR SUPPORTING THE INTEGRATION PROCESS</th>
<th>4. NEIGHBOURHOOD BASED CONFLICT RECOGNITION</th>
<th>5. PATTERN BASED RESOLUTION OF INTEGRATION CONFLICTS</th>
<th>6. COMPUTER SUPPORTED INTEGRATION WITH UTILIZING USER FEEDBACK</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Preparation of source schemata</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
</tr>
<tr>
<td>2. Recognition of conflicts and commonalities between source schemata</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
</tr>
<tr>
<td>3. Resolution of conflicts and commonalities between source schemata</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
</tr>
<tr>
<td>4. Merging the source schemata and restructuring the global schema</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
</tr>
</tbody>
</table>

A comment regarding the use of the symbols (✓) and [✓] in Table 22 is required. In phase one, Preparation of Source Schemata, guideline 3-6 is marked with (✓). This means that these guidelines are used for recognition and resolution of intra-schema differences and similarities. In phase two, Recognition of conflicts and commonalities between source schemata, guideline 6 is marked with [✓]. In phase four, Merging the source schemata and restructuring the global schema, guideline 3 and 6 is also marked with [✓]. This means that the use of these guidelines is
not explicit but rather implicitly stated in Bellström et al. (2009). Therefore, Table 22 should be viewed as a refined and extended version of the contribution given in the paper. Studies and analyses conducted after publishing the paper have indicated that these guidelines can be useful in the marked phases in Table 22. Therefore, the revised version of the mapping between guidelines and phases in the given integration process should be viewed as a further refinement of the work conducted and reported in Bellström et al. (2008, 2009).

6.3 Contribution

There are two main contributions given in the third research topic: Integration of Static and Dynamic Schemata. The first is given in the method to integrate static and dynamic schemata. In the method, I include not only the generic method proposed in Bellström & Jakobsson (2006) but also the generic guidelines proposed in Bellström et al. (2008) and the method for integrating conditions and actions proposed in Bellström et al. (2009).

The generic method for designing and integrating static and dynamic database schemata (Bellström & Jakobsson 2006) is designed as a two step method divided into a modeling phase: view modeling and an integration phase: view integration. In the modeling phase the static part of the database is first designed (modeled) based on the given customer requirements. This is followed by designing (modeling) the dynamic part of the database based on both the customer requirements and the static database schemata. In the integration phase the database schemata representing the static part is first integrated into one global conceptual database schema followed by the integration of the dynamic part based on both the integrated static database schema and the designed dynamic schemata. In the end, the method should produce one global integrated database schema representing both the static and the dynamic aspects of the database.

Since the method is generic, it is possible, on a more detailed level, to adapt and apply fitting integration methods, approaches, instruments and tools. For instance, while integrating the schemata representing the static part of the database, the method described in Batini et al. (1986), comprised of the following four phases: pre-integration, comparison of the schemata, conforming the schemata and merging and restructuring the schemata could be applied. On the other hand, if the dynamic schemata are designed using only actions and conditions
(states) then the method proposed in Bellström et al. (2009) comprised of the following four phases: preparation of source schemata, recognition of conflicts and commonalities between source schemata, resolution of conflicts and commonalities between source schemata, and merging the source schemata and restructuring the global schema could be applied. To facilitate the integration method, the generic guidelines proposed in Bellström et al. (2008) could and should be applied. This gives the designers and users a flexible method not only for integrating static schemata or dynamic schemata but also for integrating static and dynamic schemata into one schema representing the database on a conceptual level.

In the second contribution, focus was put on how to model and integrate states and state changes to ensure database consistency. The second contribution consists of a method in which I not only include the general method for modeling and integrating states and state changes in conceptual database design proposed in Jakobsson & Bellström (2007) but also the specific method focusing on how to model and integrate active database rules in conceptual database design proposed in Bellström & Jakobsson (2008).

In the method proposed in Jakobsson & Bellström (2007), we addressed how to represent and integrate states and state changes in the conceptual database schemata. More precisely, this meant representing legal states and state changes in the future database as pre- and post-conditions in the conceptual database schemata. However, it should be noted that while modeling and integrating states and state changes, several states that are not part of the database might be modeled and include in the schemata. These states are not stored in the database but in the application accessing the database. For instance, transforming the database from the consistent state \textit{Order [Stored]} to the consistent state \textit{Order [Updated]} might in the schema be represented as three states and two actions (see Figure 53).

![Figure 53. State and State Changes for Database Consistency (adapted and modified from Jakobsson & Bellström (2007 p. 42))](image)
The intermediate state Order [Changed] is a state stored in the application accessing the database while the other two are consistent database states. The actions resulting in the new consistent database state might be represented and named Change Order and Update Order since these are taking place in the application accessing the database.

Representing database states and state changes as pre- and post-conditions, responsibilities of different actors (e.g. database management system, software component) triggering different actions, could also be identified in the conceptual schema. This is important because designers and users need to know who is responsible for which action resulting in a consistent databases state.

As mentioned above in the second contribution, the proposed method for representing and integrating active database rules (Bellström & Jakobsson 2008) is also included. This is an important part of the contribution since the lack of design methods for active database rules at the design level has been pointed out as a weakness in other design approaches and methods (Amghar et al. 2000; Ceri & Ramakrishnan, 1996). Active database rules are often referred to as Event-Condition-Action (ECA) rules (Goldin et al. 2004; Pation & Dias 1999; Widom & Ceri 1996). An ECA-rule checks the condition and if the condition evaluates to true, then the action is executed. An ECA-rule could be mapped to the notation given in the Enterprise Modeling approach as follows. An event in ECA is an action in Enterprise Modeling. An Action is triggering something and so is an event. A condition in ECA is a condition (state) in Enterprise Modeling. Finally, an action in ECA is an action in Enterprise Modeling.

### 6.4 Contribution and Evaluation in Relation to Design Science

By applying design science as the chosen research method and approach, the research contribution is given in the IT-artifact as such. Artifacts are constructs, models, methods and instantiations (Hevner et al. 2004; March & Smith 1995). The first contribution of research topic three is classified as a method. In design science a method:

> […] is a set of steps […] used to perform a task (March & Smith 1995 p. 257).

The task to perform in this case is integration of static and dynamic database schemata. More precisely this means integrating both the static and the dynamic
schemata into one global schema that represents both the static and the dynamic aspects of the database.

The integration method was evaluated applying both informed arguments and scenarios. Informed arguments and scenarios are descriptive evaluation methods (Hevner et al. 2004) described as follows:

Informed Argument: Use information from the knowledge base (e.g., relevant research) to build a convincing argument for the artifact’s utility

Scenarios: Construct detailed scenarios around the artifact to demonstrate its utility (Hevner et al. 2004 p. 86)

The second contribution of research topic three is also classified as a method. More precisely this means how to represent, describe and integrate state and state changes in conceptual database design to ensure database consistency. This also includes how to, in the conceptual database schema, represent and illustrate active database rules known as Event-Condition-Action rules (Goldin et al. 2004; Pation & Dias 1999; Widom & Ceri 1996).

The method for representing states and state changes in the conceptual database schema was evaluated using informed arguments and scenarios both descriptive evaluation methods (Hevner et al. 2004).
7 Conclusions and Future Research

The purpose of this chapter is to present the conclusions drawn from the work reported in this thesis. For that purpose the research questions stated in chapter 1 and contributions given in sections 4.3, 5.3 and 6.3 are revisited and once again emphasized and discussed. A second purpose is to give an overview of possible topics for future research. This chapter is therefore divided into two parts, starting with conclusions and implications followed by the discussion regarding future research.

7.1 Conclusions and Implications

In this section the conclusions of the work presented in this thesis are presented and discussed. This is done by readdressing the contributions presented in each chapter dealing with a specific research topic (see sections 4.3, 5.3 and 6.3) and placing them in connection with the research questions stated in chapter 1. This discussion may not only contribute to a broader and deeper understanding of the contributions given in the thesis, but also to clarify the implications of the reported research results.

7.1.1 Maintaining Vocabulary in Schema Integration

Two research questions were stated for the first research topic, Maintaining Vocabulary in Schema Integration, as follows:

- What is semantic loss in connection with schema integration?
- How can semantic loss be avoided in schema integration?

The two main contributions presented in research topic one are:

- Analyzing, clarifying and explaining what semantic loss is and distinguishing between concept name compression, information loss and semantic loss in connection with schema integration.

- Proposing and demonstrating resolution methods for similarities and differences, recognized between two source schemata, which do not cause semantic loss.

The conclusions of the first research topic have two implications. Firstly, having knowledge about the problem of semantic loss and that it might occur in schema integration should make semantic loss an important topic to consider.
Designers, even users in general, must be aware of the problem and not treat it as a superficial issue that does not have to be considered in schema integration. If semantic loss occurs, it also results in several other issues such as interpretation problems.

Secondly, applying the proposed and demonstrated resolution methods for similarities and differences between two source schemata counteracts the occurrence of semantic loss. This is the case since the proposed and demonstrated resolution methods are not founded on renaming the concept names but instead to introduce and apply more precise concept names that include the original concept names.

If semantic loss is treated as a superficial problem, designers and users might resolve similarities and differences between two source schemata without taking into account the consequences of using a specific resolution method. For instance, renaming concepts could impoverish the vocabulary used in the schemata and at the same time cause semantic loss. Applying the proposed and demonstrated resolution methods might instead enrich the vocabulary used in the integrated schema. This should be viewed as a strength of the proposed and demonstrated resolution methods because concept names are retained even after resolving similarities and differences between two source schemata.

7.1.2 Integration of Static Schemata

One research question, focusing on two distinct parts, recognition and resolution of similarities and differences, was stated in the second research topic, Integration of Static Schemata:

- How can similarities and differences between two source schemata be recognized and resolved in schema integration?

The two main contributions presented in research topic two are:

- Proposing and demonstrating a method to recognize not only similarities but also differences between two compared source schemata.

- Proposing and demonstrating resolution methods for similarities and differences recognized between two source schemata.
The conclusions of research topic two can be described and explained as follows. *Firstly*, by applying the proposed and demonstrated method to recognize similarities and differences between two source schemata, several problems, such as synonyms and homonyms, might be recognized that otherwise could pass unnoticed.

*Secondly*, by applying the proposed and demonstrated resolution methods for similarities and differences between two source schemata, we are not only able to retain the original concept names by introducing more precise concept names, but also to semantically improve the quality of the schemata by introducing more precise dependencies between two concepts.

In other words, by applying the proposed method during the second phase of the schema integration process, *comparison of the schemata*, it is possible to, step by step, identify not only similarities but also differences between two source schemata that otherwise might pass unnoticed. More precisely, this means that we are able to recognize similarities and differences and thereby improve the quality of the integrated schemata. The method could also be viewed as an algorithm that the designer and users should walk through to check for similarities and differences between the two source schemata. By applying the proposed resolution methods in the third phase of the schema integration process, *conforming the schemata*, concept names are retained even after resolving similarities and differences between two source schemata. In addition, more precise dependencies might also have been introduced between two concepts. Retaining concept names and introducing more precise dependencies are two strengths that might counter the occurrence of concept name compression and semantic loss. However, it should be noted that the integrated schema should be viewed as implementation independent to be used on an implementation independent level. Viewing it from an implementation perspective, focusing on the implementation level, it might be viewed as having redundancy.

### 7.1.3 Integration of Static and Dynamic Schemata

In the third and last research topic, *Integration of Static and Dynamic Schemata*, two research questions taking both a broader and a more narrow perspective of schema integration, in connection with static and dynamic schemata, were stated. The research questions were:

- *How can static and dynamic schemata be integrated?*
• How can legal states and state changes in the future database be illustrated and integrated during conceptual database design?

The two main contributions given in research topic three are:

• Proposing and demonstrating a generic method for integrating static and dynamic source schemata into one schema.

• Proposing and demonstrating a method on how to illustrate and represent legal states and state changes in the future database during conceptual database design.

The conclusions of research topic three can be described and explained as follows. Firstly, by applying the proposed and demonstrated method, designers and users are able to first design the static schemata and then design the dynamic schemata. This is followed by integrating the static source schemata, moving on to integrating the dynamic source schemata. The method ends with a check for inconsistent use of concept names and dependencies and resolving these problems, if they exist, in the integrated schema.

Secondly, by applying the proposed and demonstrated method to illustrate and describe legal states and state changes in the database, we are able to not only illustrate the legal states and state changes but also to illustrate and demonstrate active database rules on a conceptual level. Active database rules, also known as Event-Condition-Action (ECA) rules, should be designed in a systematic way, in a method, and not in an ad hoc manner while implementing the database and information system.

In other words, by applying the proposed method, it is possible to divide the task of designing and integrating conceptual schemata into several smaller tasks. At the same time, it is possible to first put focus on the static source schemata, followed by the dynamic source schemata and in the end check for inconsistencies within the integrated schema. By applying the proposed and demonstrated method on how to illustrate and represent legal states and state changes in the database, we are able keep track of future legal database states. By adding actors we are also able to identify responsibilities, exploring which actor (database management system, software component) is responsible for which action and state, and how the database and information system should
proceed to transfer the database from one consistent state to another new consistent state. Finally, by designing the behavior of the database using actions and conditions (pre- and post-conditions), we are also able to illustrate and represent Event-Condition-Actions rules in the conceptual database schema.

7.2 Future Research

The focus of this section is future research and is therefore divided into three parts. Part one addresses work in progress, part two addresses collaboration in progress and part three addresses other topics that I find interesting for future research. At the same time, this chapter should be viewed as starting with very concrete thoughts on work in progress moving on to thoughts indicating what could be interesting and chosen for future research.

7.2.1 Future Research – Work in Progress

Including and applying the static instance-of dependency in the Enterprise Modeling approach makes it possible to integrate source schemata on two levels of abstraction (e.g. model level and instance level) into one schema. The use of the instance-of dependency was studied in connection with recognition and resolution of homonyms (Bellström 2005), inference rules (Bellström 2006b) and semantic loss (Bellström 2009). Nevertheless, how to treat the instance-of dependency in schemata integration still needs to be focused on and researched. The motivation for this is that integrating two levels of abstraction is not only complex but also important. The instance-of dependency is a powerful feature of the Enterprise Modeling approach and applying the dependency makes it possible to bridge some of the issues raised by Hoppenbrouwers et al. (2005a):

When discussing models with stakeholders and informants, in particular when trying to establish a common understanding, it is sensible to discuss different scenarios and alternatives to the model being considered. Doing so leads to an exploration of the meaning and impact of the model taking shape, and also leads to improved mutual understanding. (p. 271)

The statement given by Hoppenbrouwers et al. (2005a) throws even more light on the importance of incorporating the instance-of dependency while designing and integrating conceptual database schemata for different users since, by using the instance-of dependency, is possible to illustrate and

 [...] discuss different scenarios and alternatives to the model being considered. (Hoppenbrouwers et al. 2005a p. 271).
Discussing different source schemata with the users is important not only during schema design but also during schema integration. However, when conducting schema integration, the instance-of dependency raises new research questions, possibilities and problems that need to be addressed. Mixing data and metadata (cf. instance and model) within one schema has been addressed by other researchers. For instance, He and Ling (2004) studied the problem using the Entity-Relationship Modeling language. In the paper, the authors instead called the problem ‘schematic discrepancy’ and explained it as follows:

[… the same information is modeled as data in one database, but metadata in another. (p. 245).

Semantic discrepancy is also a problem that often arises in practice (Krishnamurthy et al. 1991; Miller 1998) and therefore more research is needed.

One specific application of the instance-of dependency is given in its strength to illustrate power types. More precisely, this means solving the problem of illustrating one important issue in one schema as an associated concept and in another schema as specialization (inheritance). Martin and Odell (1998) stated that

A power type is a type whose instances are subtypes of another type. (p. 252).

The problem of power types is also addressed in the Unified Modeling Language Superstructure Specification, v2.2 (Object Management Group 2009) where it is defined as

[… a class whose instances are subclasses. (p. 55).

Using the notation in the Enterprise Modeling Approach it is possible to semantically enrich the schemata showing, in one schema, that a concept can be both an instance-of one concept and at the same time be a specialization of another concept. Let us consider the example illustrated in Figure 54, Figure 55, Figure 56, Figure 57 and Figure 58.
In Figure 54 Report Publication, Conference Publication and Journal Publication are all specializations of Publication. This means that they all inherit Publication, meaning they are all sub-types of Publication. In other words Report Publication, Conference Publication and Journal Publication are different types of Publication(s). In Figure 55 the Type of a Publication is instead illustrated using the Type concept associated with the Publication through a dependency defined as (0, *, 1, 1).

In Figure 56 both ways to illustrate a specific Type of Publication, also illustrated in Figure 54 and Figure 55, are shown. This means that the legal values for Type are Report Publication, Conference Publication and Journal Publication. In other words, in Figure 56 the instances of Type are sub-types of another Publication (type), meaning Type is a power type. A power type is a problem that must be resolved during integration of two conceptual database schemata because otherwise the problem will surely be passed on to developers implementing the information system. In the end, it could also result in an inconsistent database since the “same” information could be stored in several places without any constraints in between them.
In Figure 57, Type is again explicitly modeled and connected to Publication via an association dependency. However, in the schema it is also shown what values are legal for Type. This means that legal instances-of Type are Report Publication, Conference Publication or Journal Publication. However, Figure 57 does not show that each Type of Publication is actually a specialization of Publication which is important to illustrate on a conceptual level. If this is not illustrated, we might miss or even lose some important dependencies in schema integration, which could also cause semantic loss.

While designing (modeling) the domain and enterprise of interest, users often express their view of the enterprise as one of the schemata illustrated in Figure 54, Figure 55, Figure 56 and Figure 57. The question that arises now is how to integrate the four schemata and at the same time not only retain all concepts, concept names and dependencies, but also how to semantically enrich the integrated schema. The answer is to integrate the schemata into the schema illustrated in Figure 58 where Report Publication, Conference Publication and Journal Publication are all modeled as subtypes (inheritance) to Publication and at the same time as instances-of Type.
Referring to the definition of power types, this means that Report Publication, Conference Publication and Journal Publication are the possible instances of Type and the same time the possible subtypes of Publication.

Finally, the instance-of dependency is not only a powerful dependency but also a very useful one. However, when including it into the schemata and putting it into the context of schema integration, it needs to be studied in more depth. The motivation for this is that we are not only integrating different levels of abstraction (model and instance level) but also since a new and powerful dependency is introduced. The instance-of dependency might semantically enrich the integrated schema but also increase the complexity of schema integration.

7.2.2 Future Research – Collaboration in Progress

During my PhD studies I have had three very interesting and fruitful collaborations which I am very grateful for. In the beginning of my PhD studies I collaborated with Sten Carlsson from Department of Information Systems Karlstad University. This was followed by a collaboration with Lars Jakobsson from Department of Information Systems Karlstad University and then a collaboration with Jürgen Vöhringer from Institute of Applied Informatics University of Klagenfurt Austria. The three collaborations, with the three research colleagues, are not sequential in time but more iterative and overlapping. This is also illustrated in the dates of publication resulting from the research work conducted together.

In this section I give an overview of future research with focus on collaboration in progress focusing on the three mentioned research collaborations.
Conducting research together with different researchers with different backgrounds and fields of expertise not only enriches the research conducted but also raises new challenges and research questions. Therefore, for me it is very important to nurture the collaborations I have built up during my PhD studies.

At the time of writing, the collaboration with Sten Carlsson can be positioned in a planning phase. We have discussed how to continue our collaboration but currently we only have rough ideas to follow.

The current status of the collaboration with Lars Jakobsson is that we are now focusing on the contributions reported in each of our joint conference papers aiming to refine, explain and clarify these and publish them in a journal article.

The collaboration with Jürgen Vöhringer is currently the most active one and has resulted in several papers presented during the time of writing the introduction chapters (cover) and therefore not included in the thesis. The main reason they are not included is because the papers were published after I decided which publications should be included. Coming from different disciplines, me from Information Systems, and Jürgen Vöhringer from Computer Science, has not only strengthened but also enriched our collaboration giving rise to several publications and interesting research discussions.

Currently, our collaboration is focusing on semi-automation of modeling language independent schema integration (see for instance Bellström and Vöhringer 2009). This is a natural extension of our research collaboration since in Bellström et al. (2009) we stated that

> Our goal is to develop a computer aided semi-automatic integration approach that supports designers and domain experts during the integration process. An important feature of our proposed strategy is its modeling language independence. (p. 21)

We have also designed an experimental study for evaluating user feedback on pre-design models (see Vöhringer et al. 2010). Evaluating user feedback is a research topic we are planning to continue with as well as to conduct several experiments aiming to collect both qualitative and quantitative data for future analysis.
In addition to these research topics, we also plan to go further into the implementation of our semi-automatic approach for modeling language independent schema integration. In connection with that, we plan to develop a prototype in which we will study and discuss the Human Computer Interaction (HCI) aspects of system design. This is an important aspect since we need to develop an easily usable graphical user interface for communicating with the users. Users are an important source of domain knowledge and should be included in the integration process because full automation is neither possible nor realistic due to its complexity (Stumpfner et al. 2004).

Finally, we are currently discussing and planning an empirical research study based on the success of our guidelines and integration process (see Bellström et al. 2008, 2009).

7.2.3 Future Research – Other Interesting Topics

In the two former sections I discussed and elaborated on several more concrete topics of work in progress and future research. In this section, I will instead discuss two other interesting topics for future research. The characteristic of these topics is that they are interesting topics of future research and I have therefore not formulated any concrete research questions or project directions. The topics are:

- Semi-automated schema integration applying the notation of the Enterprise Modeling approach as the chosen modeling language.
- Guidelines for mapping the notation of the Enterprise Modeling approach to the Unified Modeling Language.

Semi-automated schema integration applying the notation of the Enterprise Modeling approach as the chosen modeling language. Semi-automation of modeling language independent schema integration is an interesting and challenging research topic currently being studied (see for instance Bellström & Vöhringer (2009)). One interesting extension of that research topic would be to adapt and modify parts of the work in Bellström et al. (2008, 2009) using the Enterprise Modeling approach as the chosen modeling language. One possible starting point could be to use antonym relationships between verbs to aid in the process of identifying actions and interaction loops that are still missing in the integrated schema (see for instance Buchholz et al. 1997; Bounif et al. 2007). Applying the Enterprise Modeling approach is rather important since one of the goals is to
define a complete, coherent and consistent specification, schema, of the future information system (Gustas & Gustiené 2004).

One example of the applicability of antonym relationships between verbs is, for instance, while describing, illustrating and representing a library system. The two most central actions in a library are lending and returning books. If the antonym relationship between lending and returning in the library domain is stored and described in a repository, e.g. ontology, a computer-based application could indicate if one of these actions is not included in the integrated schema. This could indicate that the specific action and interaction loop, lending or returning, is missing and needs to be added.

Guidelines for mapping the notation of the Enterprise Modeling approach to the Unified Modeling Language. Even though the Unified Modeling Language has been criticized for its shortcomings (see for instance Hendersson-Sellers 2005; Jakobsson & Gustas 2004; Siau & Lee 2004), the Unified Modeling Language is today the most widely used approach for requirements engineering (Kaindl 2005). For object oriented modeling and modeling of software components it has also been mentioned as a de facto standard (Cheesman & Daniels 2001; Siau & Lee 2004). To apply the Enterprise Modeling approach on the implementation independent level, we need guidelines and heuristic rules on how to map and transform Enterprise schemata to an implementation dependent level such as the one represented in the Unified Modeling Language.

Some initiatives to combine and clarify classes in the Enterprise Modeling approach using the Unified Modeling Language have been conducted (e.g. Gustas & Jakobsson 2004; Jakobsson & Gustas 2004). However, in these publications the Unified Modeling Language is used to express classes on a level closer to implementation, meaning that the schemata expressed using the Enterprise Modeling approach are not totally mapped and converted to the Unified Modeling Language. Instead, the Unified Modeling Language is used to describe an implementation perspective of some specific classes.

Therefore one interesting topic for future research could be to start from the work presented in Gustas & Jakobsson (2004) and Jakobsson & Gustas (2004) to develop guidelines and heuristic rules for mapping and generating Unified Modeling Language schemata from the integrated Enterprise Modeling schema. Moving from an implementation independent level, using the Enterprise
Modeling approach, to an implementation dependent level, using the Unified Modeling Language, is a natural path to follow. The motivation for this is that we need to communicate the description (schema) of the future information system to several different types of users such as end users, programmers etc. By applying the Enterprise Modeling approach on the implementation independent level, it is possible to check for inconsistencies in and between static and dynamic schemata. Having done that check, the next step would be to map and generate implementation dependent schemata in the Unified Modeling Language resulting in schemata already checked for inconsistencies. The described method could be one way to bridge the shortcomings of the Unified Modeling Language pointed out by Gustas & Jakobsson (2004) and Jakobsson & Gustas (2004).

7.3 Summary of Future Research – Towards New Challenges

In the former sections I discussed and described several interesting problems and paths to follow for future research. The first section, Future Research – Work in Progress, described the most concrete proposal for future research. The second section, Future Research – Collaboration in Progress, describes status of the research collaborations I have built up during my PhD studies. The collaboration with Jürgen Vöhringer is currently the most active one and, as indicated, we have continued doing research and writing papers during the period of writing the introduction chapters (cover) in this thesis. Finally, in the third and last section, Future Research – Other Interesting Topics, I gave an overview of two other interesting topics for future research: Semi-automated schema integration applying the notation of the Enterprise Modeling approach as the chosen modeling language and Guidelines for mapping the notation of the Enterprise Modeling approach to the Unified Modeling Language. It should be noted that these two topics are currently classified as interesting topics and therefore I have not looked into them any deeper. Hence they should be viewed as just other interesting topics for future research.

Towards new challenges!
References


165


