INTEGRATION OF ENTERPRISE MODELING AND MODEL DRIVEN DEVELOPMENT – A META-MODEL AND A TOOL PROTOTYPE

Iyad Zikra
Integration of Enterprise Modeling and Model Driven Development
A Meta-Model and a Tool Prototype

Iyad Zikra
to Fatima, Osama, and Iyhab
Abstract

The use of models for designing and developing Information Systems (IS) has changed in recent years. Models are no longer considered a peripheral documentation medium that is poorly maintained and often neglected. Rather, models are increasingly seen as essential parts of the final product—as central artifacts that drive and guide the development efforts. The knowledge that modelers rely on when designing models is represented as formal models and clearly defined rules for transforming the models. The flexibility, reliability, and effectiveness offered by the formal models and the transformations are making Model Driven Development (MDD) a popular choice for building IS. Models also serve in describing enterprise design, where enterprise-level models capture organizational knowledge and aid in understanding, improving, and growing the enterprise. Enterprise Modeling (EM) offers a structured and unified view of the enterprise, thereby enabling more informed and accurate decisions to be made.

Many MDD approaches have been proposed to tackle a wide range of IS-related issues, but little attention is being paid to the source of the knowledge captured by the IS models. EM approaches capture organizational knowledge and provide the necessary input and underlying context for designing IS. However, the results produced by EM approaches need to be manually analyzed by modelers to create the initial MDD model. This interruption of the MDD process represents a gap between enterprise models and MDD models. Limited research has been done to connect EM to MDD in a systematic and structured manner based on the principles of model-driven development.

This thesis proposes a unifying meta-model for integrating EM and MDD. The meta-model captures the inherent links that exist between organizational knowledge and IS design. This helps to improve the alignment between organizational goals and the IS that are created to support them. The research presented herein follows the guidelines of the design science research methodology. It starts with a state-of-the-art survey of the current relationship between MDD and prior stages of development. The findings of the survey are used to elicit a set of necessary properties for integrating EM and MDD. The unifying meta-model is then proposed as the basis for an integrated IS development approach that applies the principles of MDD and starts on the enterprise level by considering enterprise models in the development process. The design of the meta-model supports the elicited integration properties. The unifying meta-model is based on the Enterprise Knowledge Development (EKD) approach to EM. A prototype tool is developed to support the unifying meta-model, following a study to choose a suitable implementation environment. The use of the unifying meta-model is demonstrated through the implemented tool platform using an example case study, revealing its advantages and highlighting the potential for improvement and future development.
ACKNOWLEDGMENTS

The road to this thesis was filled with the support and encouragement of many people. I thank you from all my heart.

The valuable advice of my supervisors Docent Janis Stirna and Docent Jelena Zdravkovic guided me in my work. Their continued support and feedback, patience and understanding, but most of all thought-provoking discussions taught me to think critically, reason better, and look for true meaning in what I do. I am lucky they are my advisors, and to them I extend my gratitude.

I am grateful to Professor Anne Persson. She has been familiar with my work since its beginnings, and her insights and sharp observations are always enlightening. I am also grateful to Professor Benkt Wangler, whose keen eye helped elevate the quality and presentation of this thesis.

I thank all my colleagues at DSV. Professor Paul Johannesson, I admired you since I was a master student. Your wisdom and insights guided me, especially in the beginning of my studies. I also thank Gustaf Juell-Skielse, Martin Henkel, Maria Berglotz, Eric Perjons, and Tharaka Ilayperuma for the engaging discussions and valuable feedback.

My gratitude extends to my fellow PhD students at DSV, the discussions with whom are always interesting. Constantinos Giannoulis and Eric-Oluf Svee, thank you for the insights, feedback, and ever intriguing discussions.

Working closely with the kind and generous people at the PROS research center in Universitat Politècnica de València, led by Prof. Oscar Pastor, was an inspiring step along the way. I especially thank Sergio España and Marcela Ruiz.

My friends and family, you keep me strong when the going gets tough. Thank you for being part of my life. Uncle Maher, you helped me start this journey. Your kindness and affection are ever present in my mind. My dad Osama and brother Iyhab, your love and care made me who I am. You are the beacon in my life. I love you and thank you, always.

My mom Fatina, I wish you could be with me now.
CONTENTS

1. INTRODUCTION ............................................................................................................21
   1.1. MOTIVATION ...........................................................................................................21
   1.2. RESEARCH QUESTIONS .........................................................................................22
   1.3. PUBLICATIONS ......................................................................................................23
   1.4. DISPOSITION .........................................................................................................25

2. THEORETICAL BACKGROUND ......................................................................................27
   2.1. MODEL DRIVEN DEVELOPMENT ...........................................................................27
   2.2. ENTERPRISE MODELING ......................................................................................30
   2.3. REQUIREMENTS ENGINEERING ............................................................................33
   2.4. SUMMARY ............................................................................................................37

3. RESEARCH METHODOLOGY .........................................................................................39
   3.1. DESIGN SCIENCE RESEARCH ..............................................................................39
       3.1.1. Conformance to the Design Science Research Guidelines ..............................41
       3.1.2. Implementing the Design Science Research Process ........................................43
   3.2. SYSTEMATIC LITERATURE REVIEW ....................................................................46
       3.2.1. Specify Research Questions ............................................................................48
       3.2.2. Develop Review Protocol ................................................................................48
       3.2.3. Validate Review Protocol ..............................................................................53
       3.2.4. Identify Relevant Research .............................................................................54
       3.2.5. Select Primary Studies ....................................................................................55
       3.2.6. Assess Study Quality .......................................................................................56
       3.2.7. Extract Required Data ....................................................................................57
       3.2.8. Synthesize Data ...............................................................................................58
       3.2.9. Write Review Report .......................................................................................59
       3.2.10. Validate Report .............................................................................................59
   3.3. ETHICAL CONSIDERATIONS ...............................................................................59
   3.4. SUMMARY ............................................................................................................59

4. THE RELATIONSHIP BETWEEN MDD AND EM .........................................................61
   4.1. NLP-BASED INTEGRATION APPROACHES ..........................................................62
8. GUIDELINE-BASED INTEGRATION APPROACHES ........................................64
  8.1. Edge Integration Approaches .....................................................65
  8.2. Partial Integration Approaches ...............................................65
  8.3. Total Integration Approaches ...............................................69
4. TRACEABILITY-BASED INTEGRATION APPROACHES ..........................71
4.1. SUMMARY .................................................................................72

5. PROPERTIES OF A UNIFYING META-MODEL FOR EM AND MDD ....... 73
  5.1. DESIGN-RELATED PROPERTIES ..............................................73
  5.2. METHOD-RELATED PROPERTIES ...........................................74
  5.3. MODEL-RELATED PROPERTIES ...............................................75
  5.4. TECHNOLOGY-RELATED PROPERTIES ......................................76
  5.5. SUMMARY .................................................................................78

6. DEVELOPING A UNIFYING META-MODEL ....................................79
  6.1. COMMON COMPONENTS MODEL .............................................82
  6.2. CONCEPTS MODEL ................................................................83
  6.3. GOAL MODEL ........................................................................84
  6.4. BUSINESS PROCESS MODEL ..................................................85
  6.5. REQUIREMENTS MODEL .........................................................86
  6.6. BUSINESS RULES MODEL ......................................................88
  6.7. IS ARCHITECTURE MODEL ......................................................89
  6.8. SUMMARY .................................................................................91

7. IMPLEMENTATION OF THE UNIFYING META-MODEL ...................93
  7.1. QUALITIES OF MODEL DRIVEN DEVELOPMENT TOOLS .......93
  7.2. IMPLEMENTING THE TOOL IN METAEdit+ .............................96
    7.2.1. MDD Tool Qualities in MetaEdit+ ..................................98
  7.3. IMPLEMENTING THE TOOL IN ECLIPSE ...............................102
    7.3.1. MDD Tool Qualities in EMF/GMP ..................................105
  7.4. SUMMARY .................................................................................108

8. VALIDATION OF THE UNIFYING META-MODEL .............................111
  8.1. OVERVIEW OF THE EXAMPLE CASE ......................................111
8.2. TOOL IMPLEMENTATION ...................................................................................... 112
8.3. MODELING BUSINESS GOALS ........................................................................... 113
8.4. MODELING PROCESSES AND CONCEPTS ......................................................... 113
8.5. GENERATING A WEB SERVICE-BASED INFORMATION SYSTEM SKELETON .......... 120
8.6. FULFILLING THE INTEGRATION PROPERTIES .................................................. 128
8.7. SUMMARY ........................................................................................................... 130

9. CONCLUSION ......................................................................................................... 131
  9.1. FUTURE WORK .................................................................................................. 132

REFERENCES ............................................................................................................ 135
LIST OF FIGURES

FIGURE 1. OVERVIEW OF MDD. ................................................................. 29
FIGURE 2. THE TRADITIONAL RE PROCESS. ADAPTED FROM SOMMERVILLE (2007) ............. 35
FIGURE 3. POSITIONING THE UNIFYING META-MODEL RELATIVE TO EM AND MDD. .............. 38
FIGURE 4. THE CYCLES OF DSR. ADAPTED FROM HEVNER (2007). .................................. 40
FIGURE 5. THE DSRM PROCESS MODEL. ADAPTED FROM PEFFERS ET AL. (2008). ............. 44
FIGURE 6. ACTIVITIES OF THE DSRM PROCESS AS PERFORMED IN THIS THESIS. .............. 45
FIGURE 7. THE SEARCH TERMS AND THEIR COMBINATIONS. .......................................... 50
FIGURE 8. SELECTION AND QUALITY CRITERIA. ....................................................... 52
FIGURE 9. OVERVIEW OF THE MDD PROCESS. ...................................................... 61
FIGURE 11. THE COMMON COMPONENTS MODEL. ................................................... 82
FIGURE 12. THE CONCEPTS MODEL. ........................................................................ 83
FIGURE 13. THE GOAL MODEL. ............................................................................. 84
FIGURE 14. THE BUSINESS PROCESS MODEL. ....................................................... 86
FIGURE 15. THE REQUIREMENTS MODEL. .................................................................. 87
FIGURE 16. THE BUSINESS RULES MODEL. ............................................................ 88
FIGURE 17. THE INFORMATION SYSTEM ARCHITECTURE MODEL FOR A WEB SERVICES IMPLEMENTATION. ................................................................. 90
FIGURE 18. THE METAEdit+ WORKBENCH. ............................................................ 97
FIGURE 19. THE METAEdit+ OBJECT TOOL (A) AND PROPERTY TOOL (B). .................... 98
FIGURE 20. THE METAEdit+ GRAPH TOOL. ............................................................. 99
FIGURE 21. THE METAEdit+ MODELER FOR THE CM VIEW .............................................. 101
FIGURE 22. THE IMPLEMENTATION PROCESS OF THE UNIFYING META-MODEL USING ECLIPSE .... 103
FIGURE 23. THE GENERATOR MODEL OF EMF .................................................................... 104
FIGURE 24. THE GRAPHICAL DEFINITION MODEL OF GMP .................................................. 105
FIGURE 25. THE MAPPING MODEL OF GMP .................................................................... 106
FIGURE 26. GOAL MODEL OF REACH MORE INC .............................................................. 115
FIGURE 27. THE PROCESS AND ASSOCIATED CONCEPTS DERIVED FROM BUSINESS GOAL 3 .... 116
FIGURE 28. THE PROCESS AND ASSOCIATED CONCEPTS DERIVED FROM BUSINESS GOAL 4 .... 118
FIGURE 29. CONCEPTS MODEL FOR REACH MORE INC ..................................................... 119
FIGURE 30. MAPPING PROCESSES AND CONCEPTS TO COMPONENTS OF A WEB SERVICE ISAM ... 121
FIGURE 31. JET USED FOR GENERATING WSDL DOCUMENTS ............................................ 123
FIGURE 32. JET USED FOR GENERATING XML SCHEMAS .................................................. 124
FIGURE 33. PART OF THE GENERATED WSDL DOCUMENT FOR REACH MORE INC ................ 126
FIGURE 34. PART OF THE GENERATED XML SCHEMA FOR REACH MORE INC ...................... 127
List of Tables

Table 1. Summary of the candidate and primary studies included in the SLR. .............55

Table 2. Summary of the NLP-based integration approaches ........................................63

Table 3. Summary of the guideline-based integration approaches .................................68

Table 4. Integration properties in approaches to integrating enterprise models and MDD models ...........................................................................................................77

Table 5. Summary of MDD tool qualities as perceived in MetaEdit+ and EMF/GMP. 109

Table 6. How the unifying meta-model fulfills the integration properties. ............ 128
## ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPM</td>
<td>Business Process Model</td>
</tr>
<tr>
<td>BRM</td>
<td>Business Rules Model</td>
</tr>
<tr>
<td>CCM</td>
<td>Common Components Model</td>
</tr>
<tr>
<td>CM</td>
<td>Concepts Model</td>
</tr>
<tr>
<td>DSR</td>
<td>Design Science Research</td>
</tr>
<tr>
<td>DSRM</td>
<td>Design Science Research Methodology</td>
</tr>
<tr>
<td>EA</td>
<td>Enterprise Architecture</td>
</tr>
<tr>
<td>EKD</td>
<td>Enterprise Knowledge Development</td>
</tr>
<tr>
<td>EM</td>
<td>Enterprise Modeling</td>
</tr>
<tr>
<td>EMF</td>
<td>Eclipse Modeling Framework</td>
</tr>
<tr>
<td>GM</td>
<td>Goal Model</td>
</tr>
<tr>
<td>GME</td>
<td>Graphical Modeling Environment</td>
</tr>
<tr>
<td>IMR</td>
<td>Inter-Model Relationship</td>
</tr>
<tr>
<td>IS</td>
<td>Information Systems</td>
</tr>
<tr>
<td>ISAM</td>
<td>Information System Architecture Model</td>
</tr>
<tr>
<td>JET</td>
<td>Java Emitter Templates</td>
</tr>
<tr>
<td>MDA</td>
<td>Model Driven Architecture</td>
</tr>
<tr>
<td>MDD</td>
<td>Model Driven Development</td>
</tr>
<tr>
<td>MDE</td>
<td>Model Driven Engineering</td>
</tr>
<tr>
<td>MDSD</td>
<td>Model Driven Software Development</td>
</tr>
<tr>
<td>QVT</td>
<td>Query/View/Transformation</td>
</tr>
<tr>
<td>RE</td>
<td>Requirements Engineering</td>
</tr>
<tr>
<td>RM</td>
<td>Requirements Model</td>
</tr>
<tr>
<td>SLR</td>
<td>Systematic Literature Review</td>
</tr>
</tbody>
</table>
1. Introduction

Research in Information Systems (IS) aims at developing better and more effective systems for creating, implementing, and managing methods and techniques that exist on the intersection of organizations, people, and computer systems. Improving the interaction between those three parties and increasing the value of the developed artifacts is a major goal of IS research. In recent years, the use of models in the context of IS have has shifted from being a peripheral mechanism for documentation to an integral work artifact that needs to be well designed and maintained. The usefulness of models as means for representing, communicating, and executing knowledge is being recognized (Selic, 2003).

The use of models for developing IS has affected many research areas. Enterprise-level models are increasingly used to capture organizational knowledge for the purpose of understanding how the enterprise works and discovering possibilities for growth and means to address problems. Models are also the central component in model-driven IS development approaches, where multiple levels of modeling are formally related to create smooth transitions toward a system implementation. Despite the use of models for describing enterprises and for designing IS, the transition from business models to system design models is mostly analyst-driven. The advantages of using models on both levels are not exploited. This thesis explores the possibilities of integrating the domains of Enterprise Modeling (EM) and Model Driven Development (MDD). Starting the IS development process at the enterprise level can improve the alignment between organizational design and the IS that is created to support the enterprise.

1.1. Motivation

With the increased complexity of the tasks that IS are required to perform, researchers and practitioners alike are turning towards the systematic use of models to organize information and develop the systems. The increased abstraction level offered by various modeling approaches facilitates the communication of knowledge and the transition between the phases of development (Selic, 2003). MDD has emerged as a popular development approach because it helps in improving the quality, effectiveness, and reliability of software development (Mellor, Clark, & Futagami, 2003).
Guidelines for creating the models and deriving new models from existing ones are important components in such approaches (Kent, 2002).

However, MDD is a relatively new development approach, and many unresolved challenges still influence its adoption (France & Rumpe, 2007). One of the major issues associated with MDD is the ambiguity of the source of the knowledge that is used to design the initial model. There seems to be an interruption in the development lifecycle when MDD is used. Previous development phases, covered by Requirements Engineering (RE) and EM, produce results that need to be manually studied and analyzed by modelers to create the initial MDD model (Winkler & Pilgrim, 2010). RE deals with capturing and representing the desires of the stakeholders of the system being developed, whereas EM lies on a higher level of abstraction and addresses the representation and communication of organizational knowledge. EM provides the necessary context and underlying motivation for formulating the requirements (Pastor & Pelechano, 2013). Addressing the connection between EM and MDD will consequently cover the requirements that provide input to MDD.

Limited research has been done to connect EM to MDD in a systematic and structured manner in accordance with the principles of model-driven development. The bridging of this gap in the development process will increase the alignment between the developed IS and the intended purposes for which the IS has been developed.

1.2. Research Questions

The gap that exists between EM and MDD often leads to inconsistencies between the final system implementation and the original intentions of the system. This thesis argues and demonstrates how connecting EM to MDD will facilitate the alignment between the organizational needs and the IS that are developed to support the enterprise. The research question addressed by the thesis is:

*How can EM and MDD be integrated to improve the alignment between them and facilitate the transition from enterprise design to IS design?*

The limited understanding of the relationship between MDD on the one hand and requirements and enterprise models on the other hand has led to the formulation of Research Question 1 (RQ1):
What is the current state-of-the-art of the relationship between MDD and earlier stages of IS development as captured by RE and EM?

The lack of clearly defined criteria for integrating EM and MDD is another issue that the thesis aims to investigate. It is closely related to RQ1, since the lack of approaches that realize the integration is the direct cause of the lack of integration criteria. This led to the formulation of Research Question 2 (RQ2):

*What are the properties that need to be fulfilled by an integrated EM and MDD approach?*

The following step is to design an integrated approach that bridges the gap between MDD and EM. Research Question 3 (RQ3) was formulated as:

*What are the components and semantic relationships in an integrated EM and MDD approach that fulfills the integration properties?*

Finally, validating the proposed integration approach is necessary to demonstrate its viability and the degree of fulfillment of the integration properties. This is captured by Research Question 4 (RQ4):

*Does the proposed solution establish the missing bridge between MDD and EM?*

### 1.3. Publications

The results presented in the thesis have been partially published as papers in conference proceedings. This section lists the papers using their bibliographical information. It also illustrates how parts of the thesis are related to the papers.

This paper includes an earlier version of the literature review presented in Chapter 4, as well as the integration properties discussed in Chapter 5.


This paper includes an earlier version of the unifying meta-model, presented in Chapter 6.


This paper provides the basis for the contents of Chapter 7 of the thesis.


This paper discusses the alignment between the unifying meta-model and a communication-oriented business process modeling and requirements engineering method called Communication Analysis. Findings in this paper affected the design of the unifying meta-model, which is discussed in Chapter 6.

Other publications by the author include:


1.4. Disposition

The following chapters of the thesis are organized as follows. Chapter 2 presents the theoretical background concerning MDD, EM, and RE. Chapter 3 explores the design science as a research methodology and its application in the thesis. It also discussed the application of the systematic literature review. Chapter 4 presents the results of the systematic literature review that explored the relationship between MDD and EM. Based on the findings in Chapter 4, a list of necessary integration properties for integrating EM and MDD is proposed in Chapter 5, while Chapter 6 presents and discusses the unifying meta-model as a solution to the integration problem. Chapter 7 explores the issues related to the environments within which the unifying meta-model is implemented, while Chapter 8 presents an example case for illustrating the validity and viability of the solution. Finally, Chapter 9 concludes the thesis and highlights future research and development plans.
2. Theoretical Background

The work presented in this thesis is grounded in several distinct but related domains in IS research. It resides on the intersection of EM and MDD. The relationship between these two domains covers the requirements that provide input for designing MDD models. Overall, the thesis lies within the scope of business-IT alignment. This chapter gives a brief overview of the aforementioned domains.

2.1. Model Driven Development

Until recently, advanced programming languages like Java or C# have been sufficient for developing IS. The details of complex systems have been managed using middleware platforms, such as J2EE or .NET, and supported by programming-like notations, such as the XML-based languages. However, the increasing complexities of the tasks which IS need to perform and the growing demand for shorter development cycles have exposed the limits of technologies that are based on coding (Selic, 2003). Models are recognized as suitable candidates for increasing the abstraction level of development and for supporting the migration through different phases of development (analysis, design, implementation, etc.). They can be used to increase the quality, effectiveness, and reliability of software development (Selic, 2003). To exploit the full potential of automation, complete and functional programs should be generated from models. Furthermore, the models need to be automatically verified to ensure they correctly describe the desired information system.

According to Gašević, Djurić, and Devedič (2009), MDD refers to model-centric development approaches wherein models are essential parts of the final product. Models in MDD steer the development process and carry vital design and development information in various forms. MDD enables developers to create abstractions of what needs to be built, or of the problem domain, as opposed to abstractions of how it will be done, referring to high-level development platforms and languages, called the solution domain (Schmidt, 2006). Those abstractions, or models, help in filling the problem-implementation gap that separates the problem domain from the IS solution (France & Rumpe, 2007).
MDD is also concerned with uncovering the implicit knowledge that modelers rely on when designing models. This knowledge is encoded in clearly defined rules for transforming models. The existence of such rules would facilitate later changes to the system, e.g. fixing errors or extending the functionality (Mellor et al., 2003). This knowledge, if made explicit, can be further reused in other projects where similar models are used. Reuse can also be enhanced by further abstracting the transformation rules and expressing them in terms of meta-models, thus making them more general and better applicable to a wider range of situations.

Models need to express certain characteristics to be suitable for MDD. Completeness (Favre, 2004), executability (Henkel & Stirna, 2010), inexpensiveness (Atkinson & Kühne, 2003), and understandability (Moody & Shanks, 2003; Henkel & Stirna, 2010; Selic, 2003) are only a few of the recurring characteristics mentioned in the literature. Aside from models, typical MDD approaches include meta-models, transformations between models, a process for creating and managing the models and their transformations, as well as tools to support the process (Kent, 2002). Figure 1 gives an overview of MDD and illustrates examples of typical MDD models. MDD approaches vary in the modeling notation and the information captured in the models. Transformations enable a controlled transition between the phases of the development process. They facilitate moving from one abstraction level to another by adding or hiding details to make the design executable. An MDD approach can include several stages of transformation, or it can be limited to a single transformation step. When defined between models developed in the same language, the transformations are based on the language of the models. However, when models are developed in different languages, the transformations need to refer to the meta-models. MDD approaches have various kinds of final results, such as programming code, executable models, user interfaces, or component reconfiguration.

MDD is still considered a new development approach. Many unresolved challenges influence its adoption. For example, developing a repository of MDD models is a complex and difficult task, contributing to only a few repositories being developed. This in turn hinders MDD projects' ability to create, manage, and reuse models (France & Rumpe, 2007). In addition, code generated by an MDD tool can be less efficient than manually written code (Selic, 2003). The lack of mechanisms for incorporating model changes without regenerating the whole system affects the scalability and
suitability of MDD for large projects (Selic, 2003). Transformations based on meta-models can lead to vertically isolated modeling environments, limiting the general applicability of models and transformations, and contributing to the very problem which MDD attempts to alleviate (Mellor et al., 2003).

Figure 1. Overview of MDD.

Yet, the perceived potential of exploiting MDD has led to greater interest in the field and promoted the application of MDD principles in a wide range of domains. Some attempts exist for extending MDD to cover EM-related aspects as well. However, as Chapter 5 will show, these approaches fall short of providing full EM-MDD integration.

The Object Management Group (OMG) has proposed the Model Driven Architecture (MDA) (Mukerji and Miller, 2004) as a standard MDD-based framework for developing software. MDA comprises three levels of modeling. The topmost level is called the Computation Independent Model (CIM). It includes business-level models that aim at describing the functionality of the desired system from a business perspective. The middle level is called the Platform Independent Model (PIM). It captures the design details of the system being modeled, including which data structures are going to be used and how processes will look like. All details pertaining to the actual implementation of the system are left out of the PIM. The Platform Specific Model (PSM) covers those details. It provides additional
information that is specific to the chosen implementation environment, such as the type of database and runtime environment within which the system will operate.

In this thesis, MDD is used to refer to all model-driven approaches, including MDA-based approaches. The wider and more general scope of MDD offers a better fit for addressing the research question of the thesis. Model Driven Engineering (MDE) is another recurring term in the literature. It is also considered a synonym for MDD, since both refer to the same set of core development principles, as found in Atkinson and Kühne (2003), France and Rumpe (2007), Gašević et al. (2009), Kent (2002), Mellor et al. (2003), and da Silva, Saraiva, Ferreira, Silva, and Videira (2007).

2.2. Enterprise Modeling

Enterprise Modeling (EM) aims at creating a structured and unified view of an enterprise, enabling more informed and accurate decisions to be made. EM provides a process whereby an integrated and negotiated model describing different aspects of an organization is created. Different EM techniques address their respective problems by offering multiple perspectives of organizational design. This includes business goals, processes, concepts, actors, as well as high-level IS requirements. Many EM techniques have emerged throughout the years, presenting different views of the organization and offering a wide variety of possibilities for designing, improving, restructoring, and automating all or parts of the business in question. Persson and Stirna (2001) identify two purposes for EM: 1) developing the business: shaping the business vision and identifying the strategy and means necessary to attain it, including IS development; and 2) ensuring the quality of the business: creating a unified and shared knowledge culture, and gaining the commitment of different stakeholders.

An enterprise generally refers to a unit of well-defined activities having an expected outcome and operating within the boundaries of a set of rules. The complexity associated with enterprises gives rise to a wide range of EM techniques that seem to capture and structure different types of enterprise knowledge. In other words, the intricacy and diversity of the fabric of an enterprise requires multiple perspectives in order to accurately represent the enterprise. The combination of all the perspectives yields the integrated overall view of the whole enterprise.
Some EM approaches focus on specific parts of the enterprise. For instance, the Business Process Modeling and Notation (BPMN)\(^1\) is a standard modeling language used to represent business processes graphically in an enterprise. BPMN facilitates the management of business processes by representing business activities and flow in a business-friendly manner, thus facilitating communication between actors in the enterprise and enabling the monitoring, quality control, and improvement of business routines. Analogously, the underlying motivations that guide the activities of an enterprise are identified and organized using a variety of goal modeling approaches. Simple approaches exist for goal modeling, such as the Business Motivation Model (BMM)\(^2\), where a graph-like notation is used to capture the hierarchy and influence relationships among goals and the means that help realize them (among other concepts and relationships). Other more complex approaches also exist, such as i* (Yu, 1995), where the Strategic Dependency (SD) and Strategic Rationale (SR) models capture the organizational context (in terms of actors and their relationships) and the underlying rationale associated with each actor (in terms of goals, soft goals, tasks, and resources) respectively. Much research is being done to integrate i* with other modeling approaches in order to extend its scope to cover other perspectives of the enterprise as well as other modeling domains, including IS and software development. Other approaches exist for modeling the organizational structure, the resources, and the architecture of an enterprise, to name only a few of the possible perspectives that EM is able to offer.

Other EM approaches adopt a wider scope and provide multiple views of the enterprise. In such approaches, an enterprise model consists of a number of related sub-models. Each sub-model describes the enterprise from a particular perspective, such as the purpose of the organization, business processes, entities, and organizational structure. For instance, Marshal (2000) proposes the utilization of the different parts of the UML\(^3\) notation to model enterprises. The Knowledge Acquisition in autOmated Specification (KAOS) project (van Lamsweerde, Dardenne, Delcourt, & Dubisy, 1991) offers formal means for capturing and expressing motivation (goals), agents, objects, and operations. KAOS includes an ontology for representing objects in terms of entities, their relationships, and the events that affect them. Goals in KAOS are statements of intention that dictate how

\(^1\)\url{www.omg.org/spec/BPMN/}
\(^2\)\url{http://www.omg.org/spec/BMM/}
\(^3\)\url{http://www.uml.org/}

31
agents cooperate in performing the operations that are needed to fulfill the goals. Operations describe the input/output relationships that modify and transport objects. KAOS models are intended to be used in the context of developing IS, and they cover the path from business goals to the operations that are needed on the tactical level to fulfill those goals, all the while providing the formal basis for making design decisions and connecting the different modeling perspectives. Enterprise Knowledge Development (EKD) is another EM approach that relies on six integrated sub-models to provide a holistic view of the enterprise (Bubenko, Persson, & Stirna, 2001). It covers the organization’s vision and strategy, the business policies and rules, the business ontology and vocabulary, the procedural aspects of business operations, the organizational structure, and the technical components and needs of a supporting IS. Inter-model relationships connect modeling components across the sub-models of EKD, contributing to a holistic view that allows the tracing of decisions between the models. EKD also advocates the use of a participative approach for model development (Stirna, Persson, & Sandkuhl, 2007). The Enterprises Knowledge Meta-model (EKM) is a similar EM approach that incorporates multiple complementary sub-models to handle goals, activities, roles, resources, and IS components (Loucopoulos & Kavakli, 1999).

The multi-perspective aspect of EM is a valuable property, especially when it is followed by IS development. In conjunction with business development, EM often provides input to IS development. In fact, although the focus of EM techniques has been on the organizational level (facilitating the restructuring and reengineering of an enterprise to better align the actual operations with the vision and goals), it is often the case that the supporting IS are affected by the decisions made on the organizational level. The effects of high-level changes on the supporting systems form a popular and widely explored research topic. According to Frank (2002), the purpose of EM is to offer different complementary views of an enterprise that are generic enough to be applied to any enterprise, while at the same time offering the necessary abstractions to help with designing IS that are well aligned with the enterprise’s strategy.

Enterprise Architecture (EA) is a notion that sometimes accompanies references to EM. In EA, the enterprise is dissected along multiple axes to uncover the complex interactions where people and technology interact to fulfill the vision of the enterprise. EA provides a blueprint of the enterprise infrastructure (Gustas & Gustiené, 2005). The result is usually a framework
that addresses, among other issues, the organizational, financial, responsibility, legal, and technical issues. For instance, the Zachman Framework for Enterprise Architecture is a matrix that constitutes a set of descriptive representations that are relevant for describing an enterprise, captured along two axes: the primitive interrogatives (What, How, When, Who, Where, and Why), and the reification transformations that convert an abstract idea into a concrete instantiation (Zachman, 2008). Other examples of EAs include DoDAF\(^4\) and TOGAF\(^5\), which all offer similar capabilities to that of Zachman’s Framework, albeit with varying considerations and criteria for viewing the enterprise.

It is possible to project EM approaches, whether focusing on specific perspectives or offering a holistic view of the enterprise, onto an existing EA framework (Gustas & Gustienė, 2005). Allocating models to components of an EA helps to identify aspects that have not been addressed properly, thus enabling business analysts and modelers alike to uncover additional modeling needs and better understand and improve the enterprise. Frankel et al. (2003) demonstrate how the Model Driven Architecture is mapped onto the Zachman Framework.

### 2.3. Requirements Engineering

Proper elicitation and management of requirements are considered among the major success factors in all software development projects. At the same time, failing to identify the actual needs of the stakeholders and the inability to adapt to changes in or additions to those needs is one of the most recurring problems that obstruct software development. Researchers were quick to realize the need for managing requirements. Brooks (1987, p. 16) states that:

> “The hardest single part of building a software system is deciding precisely what to build. […] Therefore, the most important function that the software builder performs for the client is the iterative extraction and refinement of the product requirements.”

This statement continues to be true today. Ever since the time of Ross and Schoman (1977)—who identified almost 40 years ago that the existence of an adequate approach to define requirements is a critical factor to avoid major problems in systems—a process for the identification, organization,

\(^5\) [http://www.opengroup.org/togaf/](http://www.opengroup.org/togaf/)
management, and documentation of the correct set of requirements has been continuously investigated, and solutions to the emerging challenges have been developed, upgraded, tested, and improved over the following decades. Requirements Engineering (RE) has become an important and ever-growing research and application domain that covers the early stages of software development. The output of the RE process represents the requirements specification (also referred to as the requirements document), and covers all functionality and quality aspects that are going to be implemented in the final system. The efforts continue today to address critical challenges that generate novel solutions in the form of extensions and changes to parts of all of the RE process activities. Nevertheless, the general view of the RE process, illustrated in Figure 2, remains largely unchanged. This section gives a summarized overview of RE, keeping in mind that every part of the RE process represents a large research topic on its own.

This thesis adopts the definition formulated by Ross and Schoman (1977, p. 6), which views a requirement as:

“A careful assessment of the needs that a system is to fulfill. It must say why a system is needed, based on current or foreseen conditions, which may be internal operations or an external market. It must say what system features will serve and satisfy this context. And it must say how the system is to be constructed.”

Although many definitions of a requirement exist, the definition above provides the needed general view of a requirement required to answer the research questions of this thesis (section 1.2) while highlighting the intentional component of a requirement, which will contribute to the development of the solution presented in Chapter 6. The research presented in this thesis does not differentiate between business/user requirements and system/software requirements. As Chapter 6 will demonstrate, all statements that affect or influence the desired functionality and quality of an IS are considered to represent requirements, regardless of the level of granularity at which the requirement is formulated, or the level of abstraction within which the requirement lies. Considering the ability to divide requirements into smaller requirements, the adopted notion of requirements in the thesis covers all types of requirements, hence endowing the proposed solution (Chapter 6) with the needed flexibility to be applied in various types of enterprises and serve the needs of a wide range of users.

The RE process (Figure 2) starts by identifying the information sources and eliciting the requirements (Sommerville, 2007). A variety of techniques
are used to elicit requirements, the most popular of which include interviews, questionnaires, prototyping, ethnography, and the study of existing documentation. The next step includes analyzing and examining the requirements to identify any possible problems that will inevitably arise. Missing requirements, conflicts between requirements, and inconsistency in the information carried by the requirements are only a few of the issues that will arise and should be addressed accordingly. Solving problems in the requirement specifications requires the involvement and collaboration of the stakeholders. During the negotiation with the stakeholders, the problems and issues that were identified during the analysis are discussed. This leads to better alternatives to represent the requirements and aids in eliciting any additional information that has been identified as missing from the existing set of requirements.

Figure 2. The traditional RE process. Adapted from Sommerville (2007).

Requirements documentation is the next stage of the RE process. At this point, requirements are captured in written form together with the analysis and negotiation information. Every requirement is represented as a statement of a desired functionality or quality in the final system. Requirements are annotated with attributes (such as priority, cost, and responsibility) that help in managing the requirements; keeping track of changes and preparing for the design and implementation of the software. During the documentation phase, the requirements specification (or requirements document) is created. The final stage of the RE process is the validation of the requirements document to make sure that all the requirements have been elicited and that they are formulated properly and according to the needs of the stakeholders. Requirements that fail the validation tests are reconsidered, returning the RE process to previous stages as necessary. Applying the RE process in practice rarely takes a linear form. Stages usually overlap as requirements are elicited, analyzed, and negotiated, causing new requirements to pop up all the time. Changes in the
stakeholders’ needs are not uncommon either. Stakeholders are prone to be unable to correctly express their needs directly. Changes may also occur to the underlying motivation for developing the IS.

To deal with change requests to requirements in a structured and effective manner, traceability links are used to indicate how requirements are related to each other. A decision to change one requirement might have a cascading effect on requirements that describe the same or other parts of the desired system. Requirements will always change; therefore, it is essential to establish the traceability relationships in order to be able to manage the requirements effectively. Stakeholders are people, and people are naturally ambiguous. Requirements provided by the stakeholders can often change, because they are never correct from the beginning.

The traditional RE process has proven to be helpful in many IS development projects. However, there are situations where the detailed documentation and tracking of requirements are considered as burdens to the development process as a whole. As a response to the challenges facing the traditional RE process, other approaches have started to emerge. Goal Oriented Requirements Engineering (GORE) is an attempt to go beyond the direct functional and quality requirements of a desired system to uncover the underlying motivation (captured as goals) that drives the requirement statements (van Lamsweerde, 2000). A better understanding of the needs for developing the system will enable requirements engineers, system analysts, and developers, and stakeholders alike to “see” the requirements better. Major approaches include KAOS, i* (which were discussed in the previous section), and the Business Motivation Model (BMM)\(^6\). In the context of enterprises, understanding the underlying motivation for developing supporting IS represents crossing over the border of the software realm into enterprise modeling. Nevertheless, KAOS and i* are considered proper RE approaches that can be fully exploited for software development without referencing the higher-level enterprise modeling issues. The NFR Framework (Chung, Nixon, Yu, & Mylopoulos, 1999) is a variation of GORE where the notion of soft-goal is used to represent quality (non-functional) requirements of a system. By capturing the relationships between the non-functional requirements, as well as the ability to refine soft-goals into more concrete and detailed goals, non-functional

\(^6\) [http://www.omg.org/spec/BMM/1.1/]
requirements are analyzed and associated with the functional requirements that they affect.

The recent rise of agile software development methods has prompted some researchers to apply the principles of agile development to RE, leading to the development of Agile RE. The documentation needs dictated by the traditional RE process are considered to be heavyweight and are thus replaced in Agile RE by lightweight activities where requirements are implemented directly as programming code (or pseudo-code). The basic premise is that changes to requirements are inevitable, and implementing them directly as code can contribute to shorter development lifecycles and quicker release times (Balasubramaniam, Cao, & Baskerville, 2010). This type of RE demands closer collaboration with stakeholders. Following an Agile RE approach is recommended when the requirements are hard to clearly identify in the early stages of the project, or where the requirements for the system evolve from new usage patterns of the system (Balasubramaniam et al., 2010). For instance, social networking services such as YouTube\(^7\) and Facebook\(^8\) regularly derive new functionality and quality constraints from explicit end-user feedback as well as automatically detected usage patterns. This generates a constant flow of new requirements that traditional RE processes are unable to manage.

Many of the major RE approaches acknowledge the intrinsic relationship between requirements and the following stages of software development, prompting the development of approaches with a wider scope than RE. Many approaches offer the capabilities of RE while overlapping with higher (i.e. EM) and lower (i.e. software design and implementation) abstraction levels. A more thorough investigation of this type of approaches is presented in Chapter 4.

2.4. Summary

The relationship between organizational knowledge and IS design has long been identified as an important factor for business/IT alignment (Wangler and Wingstedt, 1990). However, the connection is still analyst driven without a well-defined formal basis. The work presented in this thesis addresses this issue and provides a solution to systematically connect the two domains. The unifying meta-model directly targets the integration

---

\(^7\) http://www.youtube.com/
\(^8\) https://www.facebook.com/
between EM and MDD as two complementary yet separate domains in IS development. EM primarily acts on the business level, whereas MDD is oriented towards IS. In the traditional IS development sense, RE covers the stage that precedes MDD. The requirements specification (i.e. requirements document) provides input for creating the models used to design and implement the IS. In this sense, EM subsumes the notion of requirements specification and extends it with business knowledge to provide the necessary underlying motivation for developing the IS. Figure 3 illustrates how the unifying meta-model is positioned in relation to the domains presented in this chapter. Considering the interpretation of business-IT alignment adopted by Luftman (2004), where alignment refers to harmonizing the use of IT with the business strategies, goals, and needs, and considering that EM aims at representing organizational knowledge from different perspectives (Bubenko et al., 2001), the integration of EM and MDD fits within the broader framework of business-IT alignment.

Figure 3. Positioning the unifying meta-model relative to EM and MDD.
3. Research Methodology

A methodology defines guidelines as part of a generic framework for conducting research. It addresses issues that are common to the formation of new knowledge through research endeavors within a certain domain. This chapter discusses how the design science methodology guides the research in this thesis. It also discusses the practice of systematic literature review and its implications as it is adopted in the thesis.

3.1. Design Science Research

Research in IS has traditionally followed one of two paradigms: behavioral science and design science (Hevner, March, Park, & Ram, 2004). The behavioral science paradigm is mainly employed to develop theories, principles, and laws that explain or predict the social phenomena surrounding the different lifecycle phases of IS, in a similar fashion to research carried out within the natural sciences fields. However, the creation of the artifacts that are scrutinized using behavioral science methods rely on activities that have the essential characteristic of being of a problem-solving nature, whereby artificial and innovative solutions are designed and purposefully deployed to address challenges associated with the analysis, implementation, use, and management of IS (Hevner et al., 2004). Therefore, Hevner et al. (2004) advocate the complementary use of design science methods and behavioral science methods in IS research; design science helps in creating IS artifacts to solve organizational problems, and behavioral science enables the understanding of the interactions between these artifacts on one side and the people who use them and the enclosing environment on the other.

Design Science Research (DSR) is defined as follows:

Design science research is a research paradigm in which a designer answers questions relevant to human problems via the creation of innovative artifacts, thereby contributing new knowledge to the body of scientific evidence. The designed artifacts are both useful and fundamental in understanding that problem. (Hevner & Chattarjee, 2010, p. 5).

A DSR framework is compiled by Hevner et al. (2004) to position IS research relative to both the knowledge base of the IS community and the environment where the research is applied. The inner workings of IS research and the interactions surrounding it are stressed through the
identification of three distinct cycles (seen in Figure 4): the relevance cycle, the rigor cycle, and the design cycle (Hevner, 2007). The relevance cycle identifies the necessity to be relevant and aligned to the business needs stemming from the people, organizations, and technology that exist in the environment. This context provides input for research in the form of significant problems, and establishes the acceptance criteria for evaluating the research results. Conversely, the rigor cycle underlines the need for IS research to rely on previously established foundations in the research community, which manifest themselves as proven and tested theories and methods. In return, IS research contributes enhanced or novel artifacts and solutions to the knowledge base, extending the foundation for further research. The design cycle elaborates the research activities iterated during the development of the artifact, its evaluation, and refinement criteria for improving the design. Evaluation performed within the design cycle is referred to as internal evaluation, whereas that conducted against the environment criteria represents field-testing.

Another established research method within the IS domain is action research. It has the premise that complex social systems are irreducible for intelligible study. Thus, action research focuses on introducing changes to existing processes of operation and then observing the effects of these changes and updating according to the observed results (Baskerville, 1999). Action research involves two distinct stages: the diagnostics stage, involving analysis of the social system by the researchers and the practitioners; and the therapeutic stage, where collaborative changes are introduced and the effects are studied (Baskerville, 1999). Action research is more mature and has a deeper history in IS research than that of DSR. However, the notion of design as research has gained considerable support and acceptance within the IS research community in recent years, owing to its focus on creating
(designing) new artifacts. In certain research efforts, such as the work presented in this thesis, applying DSR principles is more fitting, especially considering the stated research questions (section 1.2), which do not aim at solving problems in a specific social system but strive to provide a general solution for developing IS that are aligned with organizational goals following MDD principles. Nevertheless, Hevner and Chattarjee (2010) demonstrate the overlap between action research and DSR and highlight interesting open issues that when addressed will contribute to the establishment of a more complete research method that capitalizes on the strengths of both paradigms.

The following subsections delve further into how this thesis invokes the principles of DSR.

3.1.1. Conformance to the Design Science Research Guidelines

Seven guidelines are proposed by Hevner et al. (2004) to help researchers in understanding the requirements of effective DSR. The guidelines are not intended as prescriptive and they do not constitute mandatory steps that must be fulfilled for a research effort to be labeled as design science. Instead, they provide indicators to issues that need to be addressed in a research endeavor for it to be regarded as DSR. The way this thesis conforms to the guidelines is discussed below.

**Design as an Artifact.** The design and construction of purposeful IT artifacts that are used for solving relevant problems is the goal of DSR (Hevner et al., 2004). Therefore, it is important to clearly identify the products of a research effort. Artifacts, according to Hevner et al. (2004) include: constructs that provide the building blocks for describing an IS in the form of vocabularies and symbols; models that tie constructs together and provide abstractions and representations of IS; methods that illustrate the steps required to develop an IS through algorithms and practices; and instantiations (implemented or prototype systems) that demonstrate the feasibility of using the constructs, models, and methods for developing IS. As discussed in section 1.2, this thesis focuses on bridging the gap between MDD and EM. Three related artifacts are developed to realize this. The first artifact, presented in Chapter 5, is a list of necessary properties for integrating EM and MDD. It represents a construct that influences the design of the remaining artifacts. The method that is planned in the future to support the unifying meta-model (see section 9.1) will also rely in its design on the integration properties. The second artifact, described in detail in
Chapter 6, is a unifying meta-model that offers an integrated and holistic view of the organizational and development aspects of IS. The third artifact is a prototype tool that enables the creation and management of the proposed unifying meta-model, and is presented in Chapter 8.

**Design Evaluation.** The usefulness of the designed artifacts and their ability to solve their intended problems must be demonstrated via evaluation methods (Hevner et al., 2004). The evaluation of the unifying meta-model is threefold:

- The tool presented in Chapter 8, while representing an artifact by itself, includes an implementation of the unifying meta-model, thus demonstrating its viability.
- An exemplary scenario is discussed in Chapter 8 to demonstrate the utility of the unifying meta-model for developing IS in covering organizational (high-level) as well as development (low-level) aspects, thus establishing the missing link in the development lifecycle of IS.
- The unifying meta-model and its implementation realize a necessary set of properties for integrating EM and MDD (see Chapter 5). This argument is expanded in section 8.8 to demonstrate how the designed artifacts overcome the problems that affect existing solutions to the EM-MDD integration gap (see Chapter 4).

**Research Contributions.** Research in IS must create new knowledge that extends the knowledge base of the IS community. The artifacts developed in this research effort (see “Design as an Artifact” above) represent the contributions.

**Research Rigor.** The use of established and tested methods for conducting the research and designing the artifacts guarantee the rigor of the research. In this thesis, rigor is achieved along several axes:

- A study of the state-of-the-art of the relationship between EM and MDD is conducted at the beginning of the research effort to acquire a clear understanding of existing approaches that integrate EM with MDD. This study also enables the identification of the right problems to address in new research. The study follows the principles of systematic literature review to ensure the high quality of the results.
The unifying meta-model uses the principles of an established enterprise modeling approach, namely EKD. The advantages of using EKD to bridge the gap between EM and MDD are illustrated in Chapter 6.

The solution presented in this thesis implements the principles of MDD, a software development methodology that is growing in popularity in the IS research community. MDD offers systematic and formal means for developing software by replacing code with models as the main development product.

Design as a Search Process. Acquiring the “right” design entails several iterations of design and evaluation. This applies to the artifacts presented in this thesis. The integration properties presented in Chapter 5 are a result of several iterations of search and analysis to uncover existing approaches for integrating EM and MDD. The unifying meta-model and the accompanying tool were both initially designed and evaluated (see Chapters 6 and 8), and the evaluation results provided indicators that were used to improve the designs through many iterations.

Communication of Research. Presenting the results of DSR is part of the relevance and rigor cycles. It guarantees that new knowledge is added to the knowledge base and that practitioners are able to successfully use the artifacts to solve organizational problems. This thesis and the prototype tool represent different mediums for communicating the results of the research work. On top of that, earlier versions of different parts of the thesis were published in peer-reviewed conferences, as described in section 1.3.

3.1.2. Implementing the Design Science Research Process

A Design Science Research Methodology (DSRM) is proposed by Peffers, Tuunanen, Rothernberger, and Chattarjee (2008) to complement the DSR framework of Hevner et al. (2004). The methodology summarizes the activities that comprise DSR as described in the most influential works on design science, and the result is a nominal process of six activities that cover the stages of conducting DSR, as illustrated in Figure 5. This section walks through the activities of the DSRM process and relates them to the different parts of the thesis. A summary of how the thesis realizes the activities of the nominal process is given in Figure 6.

IS research that follows DSR principles starts with the identification of the problem that is to be tackled. A clear problem statement sets the context
for developing the artifacts that will eventually be used in the solution. Understanding the need for the solution offers the necessary motivation for conducting the research and accepting the results. According to Hevner et al. (2004), the problem must demonstrate relevance by originating from the current needs of the environment in which the solution will be implemented and used. Together, the problem identification and motivation comprise Activity 1 of the DSRM process. After the initial identification of the gap that exists between EM and MDD, a literature survey was conducted as the initial stage of this thesis to expose the standing issues concerning integrating the two domains. The results of this survey are presented in Chapter 4.

Figure 5. The DSRM process model. Adapted from Peffers et al. (2008).

Activity 2 builds on the problem identified in the first activity and defines the objectives for a solution. The focus of this thesis, as described in section 1.2, is synonymous with the objectives referred to by Peffers et al. (2008). It represents a qualitative description of a solution to the EM-MDD
integration problem in the form of a unifying meta-model that consolidates aspects of EM and MDD to account for the missing link. The rational inference required by the DSRM process to move from the first activity to the second is accompanied by the formulation of a list of necessary properties for achieving integration between EM and MDD, based on the results of the literature survey.

In fact, an earlier version of the research question of the thesis represented the instigator for the literature study, which then served as the foundation for formulating a goal more relevant to the existing state-of-the-art. Therefore, the research presented herein aims at developing a mix of a problem-centered initiation and an objective-centered solution (Peffers et al., 2008). This deviation from the nominal process is illustrated in Figure 6.

Designing and developing the artifacts that will be used to construct a solution to the identified problem are the core activity of the DSRM process, and represent Activity 3 of the nominal process. As maintained in the previous chapter, the research in this thesis produces three artifacts: the integration properties, the unifying meta-model, and the tool. However, only the last two fit the description of the third activity according to Peffers et al. (2008); the integration properties represent an emergent artifact.

Figure 6. Activities of the DSRM process as performed in this thesis.

Demonstrating the use of the artifacts in a solution and evaluating the artifact following well-defined methods represent Activities 4 and 5, respectively. The nominal process proposed by Peffers et al. (2008) includes
both of these activities, but acknowledges that only one of them is sufficient (according to previous research on design science). The implementation of the unifying meta-model (as part of the tool) and the exemplary scenario demonstrate the use of the developed artifacts. The discussion in section 8.8 illustrates how the unifying meta-model and its implementation fulfill the integration properties of Chapter 5.

Finally, the results of the research must be communicated to the IS research community and the practitioners in the environment as part of Activity 6. As mentioned at the end of the previous section, the thesis, the prototype tool, and the list of publications in section 1.3 represent the communication mediums of the designed artifacts and the research results.

3.2. Systematic Literature Review

The rigor and soundness of any research question depend on the existing body of knowledge, which is demonstrated and accumulated as a result of previous research. DSR stresses this relationship by requiring the research effort to draw from the knowledge base as part of the rigor cycle (Hevner, 2007). Systematic Literature Review (SLR) is becoming increasingly popular in the IS domain as a method for surveying previous and related research and identifying gaps, opportunities, and possibilities for extending current theories and developing new ones (Webster & Watson, 2002). The medical domain, with its various disciplines, has a history of evidence-based research that relies on systematic reviews of previously conducted empirical studies in order to aggregate and consistently analyze and present the results (Sackett, Straus, Richardson, Rosenberg, & Haynes, 2000). Other fields also employ SLR techniques, including education, sociology, and business management. When it comes to IS, the relative youth of the domain and the complexity associated with assembling works in such an interdisciplinary field, as Webster and Watson (2002) point out, has postponed the adoption of SLR as a popular choice for researchers. Although domain experts can provide useful sources of information when it comes to IS research, their contributions tend to be practice-oriented with a focused perspective. More researchers are choosing SLR as a tool to understand emerging issues or to summarize mature topics because of the increasing size of the body of knowledge to be studied and the breadth of the domain. SLR provides mechanisms to cover as much relevant knowledge as possible. Therefore, processes that describe the activities of SLR and theoretical frameworks that
emphasize the importance of surveying the existing body of knowledge are emerging. The existence of several high-profile journals that are devoted to review articles (e.g. MISQ Review, ACM Computing Survey, Computer Science Review, etc.) is another indicator of the increased interest in SLRs.

SLR is defined as “a means of evaluating and interpreting all available research relevant to a particular research question or topic area or phenomenon of interest” (Kitchenham, 2004). It must be noted though that this definition is formulated within the broader context of Evidence-Based Software Engineering (EBSE), which is a research method that provides the means for researchers to establish a connection between their work in Software Engineering (SE) and the requirements of the industry. EBSE also ensures improved dependability and increased acceptability of software-intensive systems (Kitchenham, Dybå, & Jørgensen, 2004). Nevertheless, we believe that this understanding of SLR (along with the SLR process proposed by Brereton, Kitchenham, Budgen, Turner, and Khalil (2007)) can be extended to apply to research in IS domains without undermining the intended goals of the method. Research in IS disciplines overlaps with research in Computer Science (CS) and SE in aspects such as topics of interest and research approaches and methods, according to the comparative study of research in the three fields presented in Glass, Ramesh, and Vessey (2004). The multidisciplinary nature of research within IS and the reliance on reference disciplines that range from management to cognitive psychology has lead IS research to be inclined to focus more on behavioral and organizational issues (i.e. more abstract and theoretical) rather than, as Glass et al. point out, the more technical orientation of CS/SE. But the products of IS research are not limited to the behavioral context of its focus. Producing new things (processes, methods, algorithms, and products) is not exclusively confined to CS/SE research, as Glass et al. (2004) conclude. According to DSR as defined by Hevner et al. (2004), the products of IS research are IT artifacts that include constructs (vocabulary and symbols), models (abstractions and representations), methods (algorithms and practices), and instantiations (implemented and prototype systems). Moreover, the framework presented by Webster and Watson (2002) for writing literature reviews in IS provides additional evidence that it is safe to apply SLR principles to IS research.

The process described by Brereton et al. (2007) for conducting an SLR is divided into three phases: planning the review, conducting the review, and documenting the review. These phases cover 10 stages that illustrate the
steps and activities required to perform a review that fulfills the goal and purpose listed by Kitchenham et al. (2004). The studies that constitute the subjects of the systematic review are referred to by Kitchenham et al. as the primary studies. The review itself is labeled as the secondary study. We use the notion of candidate studies to denote works that have been considered but not yet identified as primary. The following subsections describe how the 10 stages are implemented in this thesis and how the results are used to address the research questions in section 1.2.

3.2.1. Specify Research Questions

The first stage in conducting a SLR focuses on the need to scope the review study. This is established by formulating a clear and robust research question (or a set of questions) that addresses a specific research problem. The steps taken to formulate the research questions that form the basis of the SLR uncover the motivation of the research and explain the contribution of review (Webster & Watson, 2002).

In standalone SLRs, i.e. SLRs that represent complete and independent research efforts, the framing of the research question takes a considerable part of the planning phase (phase 1) of SLR. However, considering that SLR is used in this thesis in a wider context (i.e. the DSR framework), the research question addressed by the SLR is RQ1 (as described in section 1.2).

3.2.2. Develop Review Protocol

The next stage in phase 1 of the SLR process is the development of the review protocol. The purpose of concretely defining a protocol is to minimize the bias of the review (Brereton et al., 2007) and guarantee that the results of the review are reproducible, hence contributing to the robustness of the research. The protocol represents the blueprint that guides the implementation of the systematic review, including the steps to be followed to conduct the review, the boundary conditions, the sources for finding candidate studies to be included in the review, the conditions to apply when selecting the primary studies (including search terms, quality conditions, and other applicable measures), and criteria and methods for analyzing the collected data and structuring the results.

Key Concepts

The important concepts that drive the selection of the primary studies are derived from the research question that underpins the SLR. The concepts and their synonyms serve as search terms (also referred to as keywords, key
terms, or search terms) that can be arranged in various combinations using the logical operators AND and OR when searching in online sources.

Understanding the relationship between existing MDD approaches and EM approaches, including those that focus on requirements, is the purpose of conducting this SLR. Therefore, MDD is a central concept that drives the selection of the primary studies. MDD has no universally accepted definition across the research community. It rather has a description of characteristics that distinguish approaches that are labeled as model driven. MDD is sometimes referred to in the literature as Model Driven Engineering (MDE) or Model Driven Software Development (MDSD), so these concepts are considered synonyms for MDD and thus used also as search terms. A popular representative MDD approach is the Model Driven Architecture (MDA), and the abundance of research on MDA qualifies it to be included as a synonym for MDD in this study. A more thorough discussion of MDD and its variants is presented in Chapter 2.

Traditionally, MDD approaches cover the design and implementation phases of the software development lifecycle. Previous stages of the development are considered to be part of Requirements Engineering (RE) or Enterprise Modeling (EM), leading to the inclusion of RE and EM as key concepts in the SLR. The concept of “requirements” is consequently important and must be considered when selecting primary studies, especially that many publications focus on requirements without referring to an encompassing RE methodology. Similarly, “enterprise models” is considered a key concept, along with representative EM techniques including “goal modeling,” “business process modeling”, and “conceptual modeling.” A deeper discussion of the interpretation of EM and RE adopted in this thesis is presented in sections 2.2 and 2.3.

The connection between EM and MDD lies at the meeting point of two levels of abstraction. EM is business-oriented, relying on models to give structure and (varying levels of) formality to organizational knowledge. Nevertheless, some approaches tend to be more loosely defined, relying in many cases on natural language to represent information. At the same time, MDD deals with specifications on more concrete and technical levels, focusing on formal models, model-to-model and model-to-text (semi)-automatic transformations, and ultimately software implementation. The relationship between MDD and EM is not limited to logical operators. Conceptual relationships such as alignment, integration, and traceability are commonly used in the literature to describe the connection between the two.
They represent applicable search terms that can refine the search results and uncover additional candidate studies. Figure 7 illustrates the search terms that are used in this SLR and the (logical and conceptual) ways in which they are combined to form search strings.

![Diagram of search terms and their combinations]

Figure 7. The search terms and their combinations.

**Publication Outlets**

Specialized journals and conference proceedings are typical venues for finding relevant candidate studies. Research databases that aggregate and index publications from these venues represent publication outlets and offer a good starting point for conducting the survey. Combinations of the search terms are used to scan the publication outlets and identify candidate studies. The following online databases and search engines are used in this SLR to identify candidate studies: ScienceDirect, SpringerLink, ACM Digital Library, and Web of Science. The results accessed through the chosen publication outlets overlap greatly, and a special effort was required to eliminate redundancies and make sure that the same candidate study was not counted more than once.

Specialized journals in IS research and development publish the leading and most influential research when it comes to MDD and EM. The SLR includes studies that were published in journals such as *Information and Software Technology, Advances in Engineering Software, Journal of Systems and Software, Information Systems Frontiers, IET Software, IEEE Transactions on Software Engineering, Software & Systems Modeling, Information Systems Research, Information & Management, International*
Journal of Information Systems Modeling and Design, and Requirements Engineering, to name only a few.

At the same time, conferences organized by the IS community provide valuable sources of relevant publications. The studies included in the SLR were published in leading conferences that include: Conference on Advanced Information Systems Engineering (CAiSE), Practice of Enterprise Modeling (PoEM), International Conference on Information Systems (ICIS), Research Challenges in Information Science (RCIS), Requirements Engineering: Foundation for Software Quality (REFSQ), Exploring Modelling Methods for Systems Analysis and Design (EMMSAD), Model Driven Engineering Languages and Systems (MODELS), and Information Systems Development (ISD).

Selection and Quality Criteria

Searching the aforementioned publication outlets using the search terms discussed earlier generates a large number of results, including a considerable proportion of false positives. Filtering the results based on well-defined criteria is thus essential and necessary to eliminate noise and possible bias and guarantee a high degree of integrity in the results of the SLR. Figure 8 illustrates how the selection and quality criteria described in this section are used to identify primary studies.

Scoping the search space by specifying the target publication outlets is the first filter for selecting the primary studies. Publications that are accessed through outlets other than those listed in the previous section are practically eliminated by omission. Therefore it is important to carefully identify publication outlets that collectively offer the widest yet most practical (considering time constraints) view of the research landscape.

The criteria used to select the primary studies that are included in this SLR range from simple and straightforward actions to more elaborate means that require studying the contents of the candidate study. Titles of search results and the lists of keywords constitute the target of the first and simplest criterion for judging the relevance of a candidate study—if the title and the keywords do not convey that the study is addressing the relationship between EM and MDD then it is immediately eliminated.

The second criterion is based on the contents of the abstract. Abstracts provide a short summary of the contents of a publication. While some publications might carry a title that covers both EM and MDD, the actual
problem that the research tackles could be unrelated to the relationship between the two domains.

Candidate studies that make it past the first two criteria must have their content studied to conclude whether the publication qualifies as a primary study and therefore must be included in the SLR. Many candidate studies that reach this point are eventually considered in the review. Still, some publications turn out to be irrelevant for answering the research question (RQ1).

A candidate study that is deemed suitable for consideration as a primary study in the review is used to uncover other candidate studies by following backward and forward citation chains. The reference list in a primary study is browsed to find older candidate studies, which will possibly form links in the backward citation chain. Conversely, publication databases are queried for newer candidate studies that reference a study that has been regarded as primary, possibly forming links in the forward citation chain.

Figure 8. Selection and quality criteria.

Primary studies that do cover the relationship between EM and MDD are finally scrutinized for the quality of the contribution. The first quality criterion targets the actual existence of integration between EM and models in MDD. Research that only describes the relationship between specific techniques, such as requirements elicitation, and MDD techniques as
consecutive steps in a development process without specifying how the organizational knowledge is connected to MDD models is not considered suitable for inclusion in the SLR. This is by no means meant to undermine the importance of research that addresses the use of EM and MDD techniques in the context of the software development lifecycle. However, the purpose of this survey is to identify the state-of-the-art of the relationship between the two domains beyond their simple use together. The second quality criterion exposes the lack of a clear description of the transitions between different stages of modeling of the MDD lifecycle. Approaches that fail to describe the steps required to transform organizational knowledge representations into design models do not fulfill the definition adopted in this thesis for MDD, and hence do not serve the purpose of this survey. The third quality criterion focuses on the utilization of an MDD approach as the target for EM integration. A large body of work exists on transforming requirements into other kinds of models, yet not necessarily within the context of an MDD approach. The lack of explicit references to MDD principles or a specific MDD approach renders the primary study unsuitable for this survey. The fourth quality criterion targets the way in which enterprise models are defined in the study. Approaches are eliminated if they rely on design-level models to capture organizational knowledge in a manner that lowers the abstraction level by creating other representations of system design. Furthermore, capturing the specific designs of certain types of systems or platforms, such as Data Warehouses (DW) or security-centered systems, is not considered relevant in this study. Organizational design should cover the notion of stakeholder desires for a system rather than design-related properties of certain types of systems.

Research is an ongoing process. Therefore it is possible to encounter multiple publications of the same authors with overlapping content that describes successive results in the same research project. In such cases, only the most recent publication is considered, unless the publications are different to the extent that they can be seen as contributing distinct ideas in terms of the relationship between EM and MDD.

3.2.3. **Validate Review Protocol**

The review protocol forms the backbone of the SLR. It describes in detail the steps required to perform the study and enable the results to be reproducible. It is therefore important to establish the validity of the protocol and its suitability for the task at hand.
The review protocol discussed in these sections has been continuously scrutinized throughout the implementation of the different stages of the SLR and regularly updated based on the cumulative findings of the search results and data extraction and analysis. The search terms and publication outlets have been updated according to candidate studies acquired through backward and forward citation chains. The selection and quality criteria have been amended following the analysis of the contents of the identified primary studies. These changes affected previously identified primary studies, which demanded they be revisited and reevaluated.

3.2.4. Identify Relevant Research

Constructing search strings depends heavily on the search functionality available at the publication outlet. Some publication outlets can process complex search expressions constructed with the help of logical operators (AND and OR). Others only allow simple search expression with one connector. Some publication outlets provide specialized search functions and operators to enable more fine-grained control over the search results. Operators to determine whether two words appear close to each other in a sentence or a phrase can be used to refine the search and limit the results to those where search terms are discussed in relation to each other. However, after trying those advanced search operators, we opted for skipping them lest we miss any positive hits that might be eliminated by the strict control.

Certain publication outlets (e.g. SpringerLink) returned search results numbering in the tens of thousands for some search string combinations, and in domains that ranged from biochemistry to urban planning. This occurred when single terms such as “requirement” or “RE” were used. To circumvent this problem, specialized operators were used to limit the fields targeted by the search to the title, abstract, and keywords of the studies. This meant repeating the search three times for each of the target fields, since there is no way of searching all three fields simultaneously.

Searching through all the selected publication outlets generated 944 candidate studies, including 294 journal articles, 548 conference and workshop papers, 96 book chapters, and 6 other types of publications (such as books). Primary studies that were identified directly from the search results by applying the selection and quality criteria (see section 3.2.2) provided starting points for forward and backward citation chains. The reference list of each of the primary studies was examined to identify additional candidate studies in backward chains, and other publications that
cited the primary study were identified in forward chains. The newly identified candidate studies were also subjected to the selection and quality criteria, and studies that were eventually designated as primary were scanned to reveal additional links in the citation chains. This cycle was repeated until all primary studies that passed the selection and quality criteria, thus contributing to the fulfillment of the goal of this SLR, were identified. Citation chains generated an additional 19 candidate studies. The majority of the studies found through citation chains were already identified through direct search, which led to the number of the newly added studies being low (compared with that of the direct search).

At the end, 963 candidate studies were found both through directly searching the publication outlets and through citation chains. Out of those studies, 137 passed the first two criteria, and 54 were found to fulfill the quality criteria. After eliminating redundant studies (e.g. studies that addressed the same issue from different angles), 37 primary studies were identified as relevant and included in the SLR. Table 1 summarizes the studies (candidate and primary) covered in the survey, showing the types of the studies, their sources, and the number of primary studies that were selected after applying the selection and quality criteria.

Table 1. Summary of the candidate and primary studies included in the SLR.

<table>
<thead>
<tr>
<th>Publication Outlets</th>
<th>Journal Articles</th>
<th>Conference and Workshop Papers</th>
<th>Book Chapters</th>
<th>Other Types of Publications</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Science Direct</td>
<td>144</td>
<td>13</td>
<td>10</td>
<td>0</td>
<td>167</td>
</tr>
<tr>
<td>SpringerLink</td>
<td>79</td>
<td>241</td>
<td>84</td>
<td>0</td>
<td>404</td>
</tr>
<tr>
<td>ACM Digital Library</td>
<td>58</td>
<td>261</td>
<td>1</td>
<td>4</td>
<td>324</td>
</tr>
<tr>
<td>Web of Science</td>
<td>13</td>
<td>33</td>
<td>1</td>
<td>2</td>
<td>49</td>
</tr>
<tr>
<td>Through Citation Chains</td>
<td>7</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>19</td>
</tr>
<tr>
<td><strong>Total before applying any criteria</strong></td>
<td><strong>963</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Total after applying first two criteria</strong></td>
<td><strong>137</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Primary Studies</strong></td>
<td><strong>37</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3.2.5. Select Primary Studies

A large number of the studies found by directly searching the publication outlets are false positives; they are results that are returned by the search engine as matching the search string but are irrelevant to the SLR. Such results include publications where the search terms (e.g. “MDA” or “RE”
carry meanings other than those intended by this survey. False positives were eliminated directly based on the title of the study (and in some cases the journal/conference where the study was published). For instance, the study by Trevelyan, Wang, and Walker (2002) addresses mechanical design improvement in two-dimensional and three-dimensional simulation.

Studies that made it past the first selection criterion were checked based on the descriptions found in their abstracts, resulting in additional eliminations. Even when a study focuses on a combination of EM and MDD, it does not necessarily have to address the integration of the two but could be eliciting the requirements for creating a novel MDD approach or the construction of an enterprise model using MDD principles. For instance, Bendix and Emanuelsson (2009) develop a number of requirements for practical model merging tools.

Candidate studies that were not eliminated after considering the title and abstract were passed on to the third selection stage, where their content was studied to determine their suitability for the goals and purpose of this SLR. 121 candidate studies made it to this point from directly searching the publication outlets. Another 16 candidate studies were discovered using forward and backward citation chains, raising the total number of studies whose content was inspected to 137. Of those, some were focused on generating code directly from requirements, thus involving no MDD-related issues. Others focused on pure MDD research describing transformations between various kinds of design models. Studies of such nature were eliminated from the survey.

Some of the primary studies were written by the same author (or authors) as part of a series of publications within the same research project. In such cases, only the most recent publication was included, and the remaining studies were eliminated as primary studies.

3.2.6. Assess Study Quality

As mentioned in section 3.2.2, primary studies that fit the purpose of the SLR still need to fulfill certain quality criteria in order to be included in the survey. Some studies were eliminated despite covering both the enterprise and system design phases of IS development, and despite the system design phase being based on MDD principles. Those studies referred to the construction of enterprise models and design models as separate (yet consecutive) phases of the development process without explaining the transition from one to the other. Studies that did not include any indication
as to how MDD models were constructed from the enterprise models were eliminated from the study.

Studies that covered the integration between EM and the design phases but fell short of demonstrating how different stages of modeling led to one another were eliminated from the survey. They lacked one of the core principles according to the interpretation of MDD embraced in this thesis. In certain cases, primary studies describe the transformation of enterprise models into other kinds of analysis and design models without a clear reference to MDD principles, namely meta-models and transformations. Processing requirements to produce models is a widely researched topic, and including studies that address this issue would inflate the number of primary studies to an impractical level that is hard to analyze and manage. Hence, studies that did not explicitly refer to MDD principles or a specific MDD approach were eliminated from the survey.

Some of the primary studies apply MDA principles for capturing requirements by constructing a business-level model (Computation Independent Model or CIM, described in detail in section 2.1). Of those, some focus on types of requirements that pertain to specific types of systems (e.g. DW) or requirements (security, Quality of Service (QoS)). These primary studies adopt an interpretation of requirements that is different from the general desires about systems that are typical of stakeholders. They were therefore considered too specialized to be included in this survey.

At the end of this stage of the SLR, 37 primary studies were left, constituting the data sources which will be analyzed as described in the next section.

3.2.7. Extract Required Data

The approaches followed by the primary studies that were uncovered by searching the publication outlets and through forward and backward citation chains were analyzed to provide an overview of the state-of-the-art of the relationship between EM and MDD. The analysis was concept-centric. The axis of the analysis evolved incrementally while the content of the primary studies was investigated.

Concept-Centric Analysis

Literature reviews need to be concept-centric in order to provide value (Webster & Watson, 2002). The concepts used to analyze the extracted data are closely related to the key concepts identified in the first phase of the
SLR. However, analysis concepts do not need to be identical to the key concepts; they can be derived from them. Webster and Watson (2002) recommend using a concept matrix to compile, analyze, and evaluate the results of the review. The matrix can then form the basis for conclusions that lead to the development of a (new or extended) theory. A model and supporting propositions can complement the theory, but one must be careful, as Webster and Watson (2002) point out, not to confuse the model and the propositions for the theory itself; the rationale that led to the development of these supporting representations constitutes the core of the theory.

The concepts used in this SLR are based on the nature of the connection established between EM and MDD by the primary studies. The form through which enterprise models are captured and represented is the first factor for constructing the analysis concepts. Elicitation techniques encountered in the primary studies ranged from text/natural language-based to model-based. The extent to which integration is achieved between EM and MDD is the second factor. Integration ranged from marginal (edge integration; where an approach is proposed for modeling the enterprise in a manner suitable for use in MDD approaches without describing the actual integration), to partial (where a transformation is proposed to convert enterprise models into models of existing MDD approaches), to total (where a novel MDD approach that natively accommodates enterprise knowledge is presented). The third factor considers establishing trace links between design models and existing enterprise models without deriving the design models from the enterprise models.

The actual results of the analysis of the primary studies, including a broader discussion of the analysis concepts, are the subject of Chapter 4.

3.2.8. Synthesize Data

The data analyzed using the concepts described in the previous section need to be consolidated in a useful and value-generating manner. Conclusions must be drawn, or improvements to amend certain deficiencies uncovered by the SLR need to be recommended. The results of this SLR demonstrate certain gaps and challenges that still hinder the full integration of EM into MDD approaches, despite the existence of a large body of research on the topic. To counter these challenges, properties that need to be realized by efforts for EM-MDD integration are proposed. Chapter 4 describes these properties, giving the motivation for considering them and illustrating how they can contribute to a more complete integration of EM and MDD.
3.2.9. Write Review Report

The results of the SLR need to be communicated by compiling a review report. The report is described as the secondary study. This and the following chapters of the thesis represent the report, since they describe the construction and implementation of the review protocol (Chapter 3) and the results of the survey (Chapter 4).

3.2.10. Validate Report

The report passed through several cycles of revision and validation before being published. As part of a licentiate thesis, the opponent and the examination committee in addition to the supervisors of the author of this thesis reviewed the report.

3.3. Ethical Considerations

The research presented in this thesis focuses on the use of formal models and transformations to capture organizational knowledge and IS design. The artifacts that are designed herein are ethically neutral because they represent abstractions (and a supporting tool) that can be utilized in any application domain. The ethical implications of a specific usage of the designed artifacts, however, are beyond the scope of this discussion.

3.4. Summary

This chapter demonstrated the method with which research in this thesis was carried out. It outlined the components of DSR and the stages of DSRM as described in the literature, while projecting those components and stages on the actual work in the thesis to illustrate how this research applies the principles of design science. The chapter also details how the thesis applies the concepts of SLR to identify and build on relevant literature.

The following chapters include the results of the research method described so far. They include the results of the SLR and the elicitation of necessary integration properties between EM and MDD. They also include the development and demonstration of the unifying meta-model, which provides a unified platform for capturing a holistic view of the enterprise and its supporting IS following MDD principles.
4. The Relationship between MDD and EM

Most existing MDD approaches focus on the details that are relevant to the development of IS. Models describe the structure of the information that will be gathered and maintained, the steps required to process the information, the constraints that govern the execution of different parts of the system, and other design and implementation related aspects of the system. MDD approaches have traditionally been designed without considering the source of the information encoded in models. Figure 9 illustrates how the creation of the initial MDD model is mostly analyst-driven. In reality, IS development is usually a part of a larger effort aimed at solving a problem within a certain context. EM and RE are two techniques that can be used to define the problem domain and understand and analyze the motivation for building the IS. Attempts to utilize the inherent relationship between EM and MDD models have recently started to emerge. Studies in this domain belong to one of three distinct groups, based on the extent to which MDD is covered and the technique used to achieve the integration. Approaches in the first group rely on Natural Language Processing (NLP) to produce a model from textual descriptions. The result thus forms the initial MDD model. Second group approaches define guidelines for directly creating the initial model from existing enterprise models. They do not rely on techniques for automatic text processing. Finally, the third group includes approaches that establish traceability links between enterprise models and MDD models (or model elements). This chapter addresses the research question RQ1 by exploring the common characteristics, benefits, and problems associated with the approaches in each group. An earlier version of the contents of this chapter was published as a conference paper (see section 1.3).

![Figure 9. Overview of the MDD process.](image)
4.1. NLP-based Integration Approaches

Using natural language for capturing requirements enables different stakeholders to understand, validate, and even actively participate in creating the requirements specification. Domain experts often prefer to use natural language descriptions rather than learn a new modeling language. However, the inherent ambiguity of natural language can cause models to differ from the real intentions of the stakeholders. Some approaches solve this problem using automatic text processing techniques to produce MDD-suitable models, replacing the analyst-driven model design.

Automatic processing of text is a large and well-established research area. However, analyzing textual descriptions to acquire requirements models that fit in an MDD process is a novel track associated with the recent rise of MDD. In this context, text can undergo syntactical analysis to find candidate concepts and relationships without delving into their meanings. Such analysis is called surface (shallow) analysis (Harmain & Gaizauskas, 2003). Semantic Analysis of text, or deep analysis (Harmain & Gaizauskas, 2003), extends syntactical analysis with additional processing of the meaning of concepts and their relationships, producing more complete models than those constructed using only syntactical analysis.

The Linguistic Assistant for Domain Analysis (LIDA) (Overmyer, Benoit, & Owen, 2001) is a methodology for creating domain models by relying on a part-of-speech tagger to process textual requirements and produce lists of candidate classes and attributes, which can be later used to populate domain models. Using only syntactical analysis, LIDA requires manual interaction with the candidate lists to graphically create the domain models. Manual validation of the resulting models is also supported by regenerating textual descriptions that can be matched to the original texts, as well as used to explain the models to non-modelers. The method presented by Montes, Pacheco, Estrada, and Pastor (2008) also uses part-of-speech tagging to decompose the text into simple sentences, but extends it with a semantic analysis stage to eliminate possible ambiguities. The result is an encoded class diagram of the future system, suitable for use with MDD. Leal, Pires, Campos, and Delicato (2008) propose supplementing traditional MDD with a controlled natural language for specifying constraints and processing sequences. The specification captured using the controlled natural language is automatically transformed and integrated with other MDD models to generate the final system.
Manually preprocessing textual requirements to allow for a more controlled processing of the natural language and increased predictability of the results (Harmain & Gaizauskas, 2003) is common to Overmyer et al. (2001) and Montes et al. (2008). Analysts in both cases must prepare use case scenarios from the original text to describe relevant operational concepts and eliminate extraneous words that are otherwise useful in defining the contexts of the concepts. The approach of Harmain and Gaizauskas (2003) tries to overcome the need for preprocessing by using a form of part-of-speech tagging that is enriched with statistical analysis and discourse interpretation. The work is supported by a tool, but is limited by being focused only on static modeling aspects. By contrast, a domain model and a set of Use Case Paths (UCP) which describe sequences of actions are the focus of the work of Ilieva and Ormandjieva (2006). They propose a four-stage approach that offers full natural language support. Text is processed using a part-of-speech tagger and arranged in tabular form. Complex statements are supported through recursive definitions of subjects, verbs, and objects. A semantic network of relationships between triads is created, whereby each triad includes a subject, a verb, and an object. A custom algorithm is used to interpret the relationships in the semantic network and produce the UCPs and the domain model.

The rapid prototype development proposed by Fu, Bastani, and Yen (2008) combines MDD with Artificial Intelligence (AI) planning and Component Based Development (CBD). MDD is used in the approach to develop the infrastructure code for the prototype, which can be reused even though the prototype itself is thrown away. To generate the infrastructure code, NLP is used to obtain a model from natural language requirements.

Table 2 summarizes NLP-based integration approaches, showing the extent of natural language coverage and the text analysis type used.

<table>
<thead>
<tr>
<th></th>
<th>Syntactical Analysis</th>
<th>Semantic Analysis</th>
<th>Combined Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subset of Natural Language</td>
<td>LIDA (Overmyer et al., 2001)</td>
<td>Sequences (Leal et al., 2008)</td>
<td>Conceptual Model (Montes et al., 2008)</td>
</tr>
<tr>
<td>Fu et al. (2008) requires a subset of natural languages, but leaves the choice of the text analysis form to the modeler.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

63
4.2. Guideline-based Integration Approaches

Many integration approaches consider an enterprise model as the initial MDD model, replacing analyst-driven model creation (see Figure 9). Unlike NLP-based approaches, guidelines are provided for creating the initial model. The extent to which the MDD process is covered varies from one approach to another. *Edge integration approaches* only discuss creating a model suitable for MDD, without details on the MDD process. Details of an actual MDD instance are not presented. To compensate, *partial integration approaches* provide guidelines for acquiring an enterprise model and transforming it into the initial model in a representative MDD process. Finally, *total integration approaches* propose complete MDD processes that are custom-built to offer native integration of enterprise-level knowledge.

Kalnins, Kalnina, Celms, and Sostaks (2010) developed a Requirements Specification Language (RSL) that combines natural language specification and process modeling. The requirements model is transformed into UML class and sequence diagrams, which are then used to generate Java code. All transformations are implemented using the transformation language MOLA. Koch, Zhang, and Escalona (2006) propose a meta-model for Web requirements, and a specific notation is derived from UML using profiles and stereotypes in order to enhance the semantics of the resulting requirements models. Another meta-model is created to describe the design of the Web application in terms of its content, navigation structure, process, and presentation. Transformations are established between the requirements meta-model and the design meta-model. Rivero, Rossi, Grigera, Luna and Navarro (2011) use MDD techniques to implement concrete User Interfaces (UI) on the Web starting from mock-ups (or initial sketches). Tags are used to label different UI components, and tag-sets semantics and heuristics are used to infer a content model that describes content flow and management. Using WebML to describe the inferred models enables their transformation into concrete UIs. The approach is supported by a tool and incorporates both MDD and agile development principles.

Table 3 summarizes the guideline-based integration approaches in terms of the modeling technique used.
4.2.1. Edge Integration Approaches

These approaches only provide guidelines for creating a model from textual requirements, and argue that such a model can be used as input to a suitable MDD process.

Some approaches employ UML-like modeling languages that facilitate integration with UML-based MDD approaches. SysML\(^9\) requirements and use case diagrams are combined to provide the requirements model (Soares & Vrancken, 2008). Executable Use Cases (EUCs) are used as a variation on traditional use cases that captures information related to the environment as well as information related to the computer-based system (Jørgensen, Tjell, & Fernandes, 2009). EUCs have three different views to allow the MDD process to start at the user-level requirements: an informal description, a formal and executable model, and a graphical representation of the executable model. UML activity diagrams are generated by Almendros-Jiménez and Iribarne (2004) starting from use cases that have been elicited using UML use case models. The activity diagrams are then used to generate UIs. Guerra-García, Caballero, and Piattini (2013) propose using a UML profile that extends use case models and activity diagrams to capture navigation and data-quality requirements for Web applications, creating the possibility for integration with the following stages of an MDD approach. Molina and Toval (2009) use a meta-model to capture requirements of Web IS with a focus on usability attributes. The meta-model is aligned with other models used in the design phase of developing Web IS, and the elicited model is used to derive the navigational model that fulfills the usability attributes. However, no clear transformation rules or guidelines are provided.

Independently of UML, a combination of tables and the Extended Graph Data Model (EGDM) is used to structure a knowledge base starting from textual requirements (Sanyal, Basu, & Choudhury, 2010). The content of the knowledge base is synthesized using the Generic Modeling Environment (GME) (Ledeczi et al., 2001) to produce a domain model.

4.2.2. Partial Integration Approaches

Minimizing the time dedicated to obtaining the final software product is the goal of all enterprise-enhanced MDD approaches. In addition to guidelines for building the initial MDD model, partial integration approaches take an

\(^9\)http://www.omg.org/spec/SysML/
extra step and illustrate how MDD can proceed in using the model. Existing MDD approaches form the basis for partial integration.

A common trend in partial integration approaches is to use a Goal Oriented Requirements Engineering (GORE) framework called i* (Yu, 1995) to produce an MDD-compliant model. Manual guidelines are proposed in (Alencar et al., 2009) for transforming the Strategic Rationale (SR) model of i* into the conceptual model of the OO-Method (Pastor, Gómez, Insfrán, & Pelechano, 2001). The SR model captures actors and their requirements, exposing the rationale for adopting the requirements and forming suitable input to MDD. However, following the guidelines requires experience in i* and knowledge about the context surrounding the actors and resources defined in the SR model. To avoid manual tasks, Martínez, Castro, Pastor, and Estrada (2003) ground the SR model in a formal specification inspired by KAOS (Dardenne, van Lamsweerde, & Fickas, 1993). Only relevant information in the SR model is considered, referring to goals that will eventually be supported by the final system. Moreover, a new “system” actor is introduced in the model to address the necessary tasks and resources needed to accomplish the relevant goals. The resulting formal SR model can be automatically transformed into the conceptual model of the OO-Method. Lucena et al. (2009) follow a similar approach, whereby SR models are automatically transformed into architectural models in the ACME Architectural Description Language (ADL). A horizontal transformation prepares the SR model by bringing it closer to the architectural model, and the resulting modified SR model is vertically transformed into the destination model.

Insfrán, Pastor, and Wieringa (2002) propose another partial integration approach to software development utilizing the OO-Method. Multiple techniques from the requirements engineering framework TRADE (Wieringa, 1999) are combined to produce a requirements model, which is then analyzed to produce a conceptual schema and associated sequence diagrams in the OO-Method, mainly through manual intervention. Similarly, the OO-Method is integrated with Communication Analysis in Pastor, Ruiz, and España (2011) to enable the capture of high-level information exchanges that describe the communication within an IS. The exchanges represent requirements, and transformation guidelines are proposed to realize the integration.

Some approaches try to extend MDD with enterprise models through different adoptions of Aspect Oriented System Development (AOSD) (Brito
Combining AOSD with MDD can provide the necessary mechanisms for separating crosscutting concerns in models (Aksit, 2005). Sánchez, Moreira, Fuentes, Araújo, and Magno (2010) propose transforming a UML model (a sequence diagram) of the textual requirements into an aspect-oriented architectural model. The original UML model adheres to a custom-built UML Profile that captures the requirements as a set of scenarios. An AOSD-based process is adapted by Fatwanto and Boughton (2008-a, 2008-b) to the special natures of both functional and non-functional requirements, by combining use cases, the NFR Framework (Chung, Nixon, Yu, & Mylopoulos, 1999), and scenario-based specification. Automatic transition to the following steps of the process is facilitated in both cases using an executable variation of UML. Fatwanto and Boughton (2008-a, 2008-b) make use of xtUML (Mellor & Balcer, 2002) to capture the models and enable automatic transition to the following stages of development.

Language Extended Lexicon (LEL) (Lindstrom, 1993) is combined with the NFR Framework by Cysneiros and Leite (2004) to model non-functional requirements in a complementary manner to other functional requirements models, including use cases, class diagrams, and sequence diagrams. Debnath, Leonardi, Mauco, Montejano, and Riesco (2008) also use LEL, together with Scenario Models (Leite, Hadad, Doorn, & Kaplan, 2000), to capture requirements, whereby the LEL provides the vocabulary for populating the scenarios. Both LEL and Scenario models are natural language-based, allowing the stakeholders to understand the documented requirements and participate in evaluating and improving them. However, a set of transformation rules (implemented in a tool) is used to process the requirements and derive an initial class diagram. Kop, Mayr, and Yevdoshenko (2007) extend the OLIVANOVA toolset, which implements MDD principles for software development, with a requirements modeling approach called KCPM. Mappings are identified between the two approaches and used to formulate transformation guidelines.

Some MDD approaches are tailored to specific areas and account for their unique characters. For example, the SR model of i* is used by Mazón, Pardillo, and Trujillo (2007) to capture requirements for a multi-dimensional (MD) conceptual model for a Data Warehouse (DW). UML profiles are designed to capture the SR model, the specific needs of DW modeling, and the resulting MD model. Query/View/Transformation (QVT) statements are then applied to the customized SR model to obtain the MD conceptual model. Similarly, the tight coupling between requirements and product line
models is identified as the major problem of existing solutions in Software Product Lines (SPLs) (Guelfi & Perrouin, 2007), where all possible variation points in a SPL need to be known beforehand, limiting the ability to derive products for requirements that are not directly derivable from the available components and architecture. Guelfi and Perrouin (2007) deploy UML diagrams, Object Constraint Language (OCL) expressions\(^{10}\), textual use cases, and a custom-built transformation language to solve this problem and help developers derive products in a flexible and coherent manner.

Wanderley, da Silveira, Araujo, and Lencastre (2012) suggest capturing creative requirements using mind maps. A meta-model for mind maps is developed and used as the source in transformation rules that generate SPL feature models.

Kalnins, Kalnina, Celms, and Sostaks (2010) developed a Requirements Specification Language (RSL) that combines natural language specification and process modeling. The requirements model is transformed into UML class and sequence diagrams, which are then used to generate Java code. All transformations are implemented using the transformation language MOLA. Koch, Zhang, and Escalona (2006) propose a meta-model for Web requirements, and a specific notation is derived from UML using profiles and stereotypes in order to enhance the semantics of the resulting requirements models. Another meta-model is created to describe the design of the Web application in terms of its content, navigation structure, process, and presentation. Transformations are established between the requirements meta-model and the design meta-model. Rivero, Rossi, Grigera, Luna and Navarro (2011) use MDD techniques to implement concrete User Interfaces (UI) on the Web starting from mock-ups (or initial sketches). Tags are used to label different UI components, and tag-sets semantics and heuristics are used to infer a content model that describes content flow and management. Using WebML to describe the inferred models enables their transformation into concrete UIs. The approach is supported by a tool and incorporates both MDD and agile development principles.

\(^{10}\) http://www.omg.org/spec/OCL/2.0/
### Table 3. Summary of the guideline-based integration approaches.

<table>
<thead>
<tr>
<th>Edge Integration</th>
<th>Partial Integration</th>
<th>Total Integration</th>
</tr>
</thead>
<tbody>
<tr>
<td><em><em>i</em>-based models</em>*</td>
<td>Alencar et al. (2009), Lucena et al., (2009), Martínez et al. (2003); Mazón et al. (2007).</td>
<td>Sketches (Rivero et al., 2011). Use Cases and Scenarios (Cysneiros &amp; Leite, 2004; Debnath et al., 2008; Fatwanto &amp; Boughton, 2008-a, 2008-b; Insfrán et al., 2002). KCPM (Kop et al., 2007). <strong>Notes: LEL is used in Cysneiros &amp; Leite (2004) and Debnath et al. (2008). NFR Framework is used in Cysneiros &amp; Leite (2004) and Fatwanto &amp; Boughton (2008-a, 2008-b).</strong></td>
</tr>
<tr>
<td><strong>UML-based models</strong></td>
<td>SysML (Soares &amp; Vrancken, 2008). EUCs (Jørgensen et al., 2009). UML (Almendros- Jiménez &amp; Iribarne, 2004; Guerra- García et al., 2013). Requirements meta-model (Molina &amp; Toval, 2009).</td>
<td>UML (Biffl et al., 2007; Guelfi &amp; Perrouin, 2007; Koch et al., 2006; Sánchez et al., 2010). Communication Analysis (Pastor et al., 2011). Mind maps (Wanderley et al., 2012). RSL (Kalnins et al., 2010).</td>
</tr>
<tr>
<td><strong>Graph-based models</strong></td>
<td>EGDM (Sanyal et al., 2010).</td>
<td>ATRIUM goal model (Navarro &amp; Cuesta, 2008). Task taxonomy (Valderas &amp; Pelechano, 2009).</td>
</tr>
</tbody>
</table>

Biffl, Mordinyi, and Schatten (2007) try to overcome the difficulties associated with reconfiguring information-sharing middleware by proposing an MDD-based approach. Information that describes contracts, policies, and capabilities is captured in explicit models. Then, the models are transformed into intermediate models of system configuration, which are subsequently transformed into actual node configuration.

#### 4.2.3. Total Integration Approaches

Some MDD approaches are specifically developed to cover enterprise-level information. Such processes stretch from the requirements stage to the execution stage, creating a comprehensive approach with native requirements support.
For example, Moros, Toval, Rosique, and Sánchez (2013) propose an MDD-compliant approach that relies on meta-models and transformations for generating home automation applications. The meta-models defined by the approach include one for requirements specification and another for inter-requirement traceability. Traceability from requirements to other artifacts in the application design model is also described using a meta-model. The requirements model is created and transformed to an application model defined by a DSL for home automation modeling. At the same time, a traceability model is generated to capture the relationships between requirements in the requirements model and artifacts in the application model. Tools accompany the proposal for managing the creation and transformation of models as well as viewing traceability information.

ATRIUM (Architecture Traced from Requirements by Applying a Unified Methodology) (Navarro & Cuesta, 2008) is an MDD methodology for supporting the architectural design decisions and rationale during AOSD. A combination of KAOS and the NFR Framework is used to capture functional and non-functional requirements in the form of a goal model, which dictates how architectural elements are synthesized to create a draft of the final system description. Architectural design choices are made based on the goal model, and architectural elements are synthesized to create a draft of the final system description. This proto-architecture is refined during the following stages of the methodology, supported by a tool that facilitates the creation and transformation of the models while exposing traceability links between them.

ProjectIT is a role/task oriented MDD approach to interactive system development (da Silva et al., 2007). Requirements are captured in a formal language and then loaded into entities and use case views, ready to be transformed into UI views using model-to-model transformations. Another transformation path from models to code relies on code templates of a skeleton of the final system, which is then finalized by a programmer. Integration with the other phases of development is supported by a tool which offers modules for modeling the requirements, designing interaction views, and generating and editing code.

The approach proposed by Valderas and Pelechano (2009) specifically supports navigation requirements of Web applications that stem from the sequential nature of the Web experience. Activity diagrams are used to describe the steps of tasks that are modeled using a hierarchical taxonomy. The taxonomy is transformed into a navigational model that indicates how
information and functionality are organized in navigational contexts and links. Viewing the requirements model and the navigation model as graphs allows graph transformation rules to be used to map them to each other, and including the input of the transformation rule and its type in the output enables traceability. Using a prototype traceability tool, the output graph can be processed to retrieve and display the traceability information in the HTML format.

Finally, UniFrame (Bryant et al., 2003) is a methodology for assembling distributed applications from heterogeneous components, where natural language specification is automatically translated into a running program. The syntactical, semantic, and pragmatic information of requirements is analyzed and stored in a knowledge base, which is then automatically transformed and integrated with other formal platform-specific knowledge to produce a platform-specific unified meta-component model. A supporting tool allows users to improve the specification and enter missing information where needed.

4.3. Traceability-based Integration Approaches

A trace can be defined as any evidence, explicit or not, between any two artifacts on the same level of abstraction or on different levels of abstraction or phases of the development process (Winkler & Pilgrim, 2010). Traceability between MDD models and previous stages of development enables a retrospective understanding of the rationale behind design decisions and facilitates change propagation. Some integration approaches are based on establishing traceability links after having created the models. The links support the analyst-driven creation of models (see Figure 9), rather than replace it.

Only two approaches were found to rely on traceability to achieve integration. A taxonomy of traceability types in SPL is used to define an MDD traceability framework in Anquetil et al. (2010). The aim is to capture the semantics of trace links, instead of a simple indication that artifacts are related. The taxonomy is generic, and can be used to trace requirements to models (or model elements). A similar MDD traceability framework is found in Ferreira and da Silva (2009), and it provides a basis for tracing the requirements and assessing their fulfillment in the intermediate models and the final product.
4.4. Summary

This chapter presented the results of the SLR of the relationship between MDD and previous stages of the IS development process. Three groups of works were identified: 1) NLP-based integration approaches, which rely on techniques for processing textual descriptions to derive MDD models; 2) Guideline-based integration approaches, which consider a requirements or enterprise model as the input to the MDD process and provide guidelines for constructing the initial MDD model; and 3) Traceability-based integration approaches, which provide traceability links between existing requirements models and MDD models. The guideline-based approaches were further divided into edge, partial, and total integration approaches based on the extent to which they support the MDD process. In the next chapter, we elicit a set of necessary properties for the integration of EM and MDD based on the findings presented in this chapter.
5. Properties of a Unifying Meta-Model for EM and MDD

The approaches discussed in Chapter 4 rely on different techniques, technologies, and methods for integrating enterprise models and MDD models. Although they share a common premise of integration, they do not agree on a core set of properties for realizing that. Based on the surveyed approaches, as well as the existing needs for rigorous techniques of model construction (van Lamsweerde, 2010), the following integration-relevant properties were elicited and organized in four general categories. They address the research question RQ2, and represent criteria that need to be fulfilled to realize EM integration with MDD. The list represents a set of necessary integration properties. It is by no means complete. Additional properties can exist to address other aspects of integration not covered in the scope of the thesis.

The surveyed approaches are summarized in Table 4 to illustrate which properties are fulfilled by each approach. An earlier version of the properties was published as a conference paper (see section 1.3).

5.1. Design-related Properties

To achieve integration between EM and MDD, any proposed approach should cover creating and structuring models that:

1. **Support requirements that define both static and dynamic aspects of the information system.** Static aspects of a system cover the relevant concepts and their relationships, and the various states through which the system passes. Dynamic aspects are the processes in the system: how they alter the static aspects of the system, and how they transform the system from one state to another. Usually, static and dynamic aspects are interleaved within the requirements, and an effort to separate them is necessary since they are spread across different models. Approaches which are NLP-based and guideline-based were split between those that only support static aspects through e.g. domain/conceptual models or class diagrams and those that support both static and dynamic aspects by including e.g. sequence diagrams or colored petri-nets (see Table 4). Static and dynamic aspects are irrelevant for traceability-based approaches.
2. **Support intentional aspects of the requirements.** Deriving models from requirements involves understanding the goals which the requirements are meant to serve, since this can lead to a better model design and hence a more fitting system. Approaches based on the i* SR model provide such intentional support, since i* is used in Goal Oriented Requirements Engineering (GORE). Also, the approach presented by Biffl et al. (2007) includes modeling contracts, policies, and capabilities, thus capturing the motivation driving the models in the following stages. Capturing the intentional aspects has particular importance when the development process spans enterprise knowledge as well, and an integrated MDD approach must include mechanisms that cover the goals relevant to the requirements, the environment from which they originate, and the environment which will surround the final system.

3. **Support requirements that define architectural aspects of the information system.** Requirements may convey information that helps in designing the architecture of the desired system. Many approaches include mechanisms to account for architectural aspects in requirements (see Table 3), such as user interfaces, ports, and configurations. Architectural information is not necessarily part of the enterprise design, but modelers should have the ability to capture it explicitly when it is present.

### 5.2. Method-related Properties

Besides design-related issues, an approach that integrates EM with MDD should cover the development process itself, namely:

4. **Include explicit model design guidelines for the initial MDD model.** Essentially, any MDD approach defines the models used for capturing different information about the intended system. When enterprise design is concerned, guidelines for acquiring the input to the initial MDD model must be present, supported by the built-in functionality of the MMD tool. Most approaches presented in Chapter 4 include such guidelines. Exceptions include the edge integration approaches of Jørgensen et al. (2009) and Sanyal et al. (2010), which explain the nature of EUC and EGDM, respectively, without detailing how they can be constructed. All traceability-based approaches belong to the exceptions group as well.
Enable change propagation between enterprise models and MDD models. Links between enterprise and MDD models enable the analyses of the effect of changes in one on the other. Navigation between models and the source of their content must be supported. Traceability can be a bi-product of the guidelines for creating the initial MDD model. Many approaches exist to establish and impose traceability links between different types of models and modeling components. For instance, Almeda, Iacob, and van Eck (2007) propose a meta-model for describing traceability between requirements and models in MDD, together with a tool for assessing conformance between requirements specification and MDD models that are derived from them. However, most integration approaches fail to capitalize on this opportunity despite the existence of model design guidelines. In addition to traceability-based approaches, only the total integration approaches presented by Navarro and Cuesta (2008) and Valderas and Pelechano (2009) offer traceability support. The former uses transformations to check the consistency of updated models or to enforce changes when possible. The latter suggests using weak and strong traceability to denote optional and mandatory elements, respectively.

5.3. Model-related Properties

The construction of models is relevant when attempting to extend MDD with EM support. The following model features should be addressed:

6. Improved model completeness. Software development can be seen as translating requirements to an application that fulfills the requirements. In MDD, models, forms, and accompanying “code” fragments such as SQL queries, are the development artifacts, so obtaining complete models is essential for obtaining software that delivers the desired results. Any integration approach must include techniques to verify the models and measure the degree of model completeness—how accurately the models represent the requirements. Only a few approaches address model completeness, and even then it is limited to manual verification by modelers or stakeholders. For example, modeling the contracts and policies in Biffl et al. (2007) enables stakeholders to offer feedback that contributes to more refined node configurations. Approaches that attribute model completeness to
the fact that a model adheres to its meta-model definition, rather than including explicit mechanisms to guarantee model completeness, are considered not to fulfill the completeness property. The total integration approach developed by Moros et al. (2013) is an example of such approaches. Model completeness complements the property for enabling change propagation; when deviations are discovered in the models, change management techniques are deployed to eliminate them and realign the models.

7. **Reusability of models.** Increasing the reusability of software components is a goal of many software development approaches and tools as well as research projects. Reusability is at the heart of Object Oriented (OO) software development, Service Oriented Architecture (SOA), and Software Production Lines (SPL), to name only a few. An EM-aware MDD approach must enable reusability of models that stem from similar settings and serve similar needs, since reusing models will increase the abstraction and granularity levels of dealing with reuse artifacts. In the case of MDD, the creation, storage, retrieval, and application of reuse artifacts is paramount because the whole development activity takes place at this level. However, achieving model reusability seems to be an open issue, since only a few of the presented approaches support it.

5.4. **Technology-related Properties**

A proposed integration approach needs the underlying technology that facilitates all the other properties. Since tools are essentially seen as the wrappers that encapsulate the technology, the approach should:

8. **Provide supporting tools.** MDD implies extensive development and use of models. In addition, some MDD approaches also require model transformations. This makes tools essential for the practical application of MDD. Integrating enterprise models with MDD models demands the support of tools. Yet, less than half of the proposed approaches provide tool that support the suggested integration guidelines. A number of those tools are research prototypes and/or are still under construction.
Table 4. Integration properties in approaches to integrating enterprise models and MDD models.

<table>
<thead>
<tr>
<th>Integration properties</th>
<th>Static and dynamic aspects</th>
<th>Intentional aspects</th>
<th>Model design guidelines</th>
<th>Architectural aspects</th>
<th>Change propagation</th>
<th>Model reusability</th>
<th>Tool support</th>
</tr>
</thead>
<tbody>
<tr>
<td>NLP-based integration approaches</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fu et al. (2008)</td>
<td>static</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Harmain &amp; Gaizauskas (2003)</td>
<td>static</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ilieva &amp; Ormandjieva (2006)</td>
<td>both</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Leal et al. (2008)</td>
<td>both</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Montes et al. (2008)</td>
<td>static</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overmyer et al. (2001)</td>
<td>static</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Edge integration approaches</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Almendros-Jiménez &amp; Iribarne (2004)</td>
<td>both</td>
<td>•</td>
<td></td>
<td>•</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Guerra-García et al. (2013)</td>
<td>both</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jørgensen et al. (2009)</td>
<td>both</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Molina &amp; Toval (2009)</td>
<td>both</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Soares &amp; Vrancken (2008)</td>
<td>static</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sanyal et al. (2010)</td>
<td>both</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Partial integration approaches</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alencar et al. (2009)</td>
<td>static</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Biffl et al. (2007)</td>
<td>both</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cysneiros &amp; Leite (2004)</td>
<td>both</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Debnath et al. (2008)</td>
<td>static</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fatwanto &amp; Boughton (2008-a, 2008-b)</td>
<td>both</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Guelfi &amp; Perrouin (2007)</td>
<td>static</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Insfrán et al. (2002)</td>
<td>both</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kalniņš et al. (2010)</td>
<td>both</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Koch et al. (2006)</td>
<td>both</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kop et al. (2007)</td>
<td>static</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lucena et al. (2009)</td>
<td>static</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
In this chapter, a set of necessary properties for integrating EM and MDD were elicited. The approaches surveyed in the previous chapter provided input for formulating the properties. The fulfillment of the properties is necessary for achieving proper integration between EM and MDD. The chapter ends with a table that illustrates the extent to which the surveyed approaches fulfill the properties. This reveals the lack of full EM-MDD integration. The next chapter presents a solution to the EM-MDD integration gap that takes into account the identified integration properties.
6. Developing a Unifying Meta-Model

Closer inspection of the current relationship between EM and MDD reveals the need for better integration. The results presented in the previous two chapters show that a variety of techniques rely on textual descriptions of the requirements using natural language or a controlled subset thereof. These techniques construct explicit relationships between requirements models and the models used in subsequent stages of the model driven software development. The requirements need to be preprocessed and pre-prepared for transformation. The resulting solution is morphed into an enforced connection rather than true integration between intrinsically related views of the enterprise and its supporting IS. On one hand, requirements are usually manipulated to fit the design models. On the other hand, the design models are not adapted to capture the information carried by the requirements (with the exception of total-integration approaches, whereby custom design models are created for the purpose of capturing requirements content).

An efficient approach to business oriented IS design and development in an enterprise is essential for gaining higher performance in product and service delivery. Estrada, Morales-Ramírez, Martínez, and Pastor (2010) point out that “Web service modeling is not considered within the organizational environment,” and this observation can be extended to other environments chosen to implement organizational design. At the same time, goals can provide the necessary contextual knowledge that defines and binds the elements of the enterprise architecture (van Lamsweerde, 2008). To cope with the complexity of the relationships between the business and IS capabilities of the enterprise, the use of an integrated architectural view that encompasses business, information, application, and technology components has been widely accepted, and manifests itself in the array of EA frameworks that have been developed. However, such views often tend to be complex and unmanageable in practice (Kaisler, Armour, & Valivullah, 2005). A viable solution needs to concentrate on essential elements and represent them in the form of models. This allows deeper insights into each of the elements and clearly exhibits the interactions between them.

The integration properties presented in Chapter 5 highlight the complex endeavor of building an integrated approach that addresses business and IS concerns simultaneously in a manageable and model-oriented manner. Attempts to create such solutions must realize those properties. In addition
to model orientation, the solution must be structured as a method with an expressive design scope. Providing tool support is also essential. An integrated enterprise architecture emerges as an aligned solution to the elicited properties of Chapter 5. Enriching it with the MDD aspects fostered by the properties would contribute to solving the problem of creating the initial MDD model, hence achieving EM-MDD integration and a better fit between the developed IS and the goals of the enterprise.

Business development and IS development overlap in terms of how they describe static, dynamic, structural, and behavioral perspectives, and capturing the overlap in models can enable a smooth and natural transition between the two abstraction plains. The business development component also covers the intentional perspective, whereas the IS development component covers the requirements and IS architecture perspectives. The integration between the business perspective (represented by enterprise models) and the IS perspective (represented by IS design models) is realized through a single unifying meta-model, thus eliminating the need for multiple levels of model abstraction—a common practice in MDD approaches—and leaving only a single transformation step towards the implementation platform. Moreover, the overlap between the two perspectives enables a seamless and continuous flow of information starting on the business level and continuing to the IS level.

The main objective for developing the unifying meta-model is to address research question RQ3 by providing a unified platform for designing enterprise models, which are then used to derive IS models that can be subsequently used to generate a functioning and complete system using an MDD approach. The unifying meta-model was subject to several cycles of development, and an earlier version of it was published as a conference paper (see section 1.3). The meta-model defines multiple complementary models, offering a holistic view of the organization and enabling automatic generation of an IS that is described by the relevant models. The meta-model defines (1) enterprise-level models, which are models that represent enterprise knowledge, as well as (2) system-level models, which describe the IS requirements and the technical components and user interfaces that are involved in the implementation of the IS. The unifying meta-model is based on EKD, an established EM approach that provides multiple perspectives of the enterprise using complementary models that capture the goals, rules, resources, processes, requirements, and technical components (Bubenko et al., 2001). EKD is selected as the candidate EM technique. It
provides the basis of developing the unifying meta-model because it includes an overall model composed of inter-related sub-models for integrating different views of the organization. Reusing EKD as an EM approach is in line with the DSR guideline of “research rigor,” which recommends among other things that the designed artifacts should be based on established and tested approaches. Furthermore, the unifying meta-model extends EKD with the principles of MDD to facilitate the transition from EM to IS.

Relationships between enterprise-level and system-level models are formalized to support evaluation of the system models and improve traceability to their origins in the organizational design. Coarse-grained relationships are used to connect the models and give a general overview of the enterprise in the form of interactions between the models (Figure 10). As the details of each model are elaborated, the coarse-grained relationships are interpreted as fine-grained relationships (also called inter-model relationships or IMRs) that relate model components across different models. They depict the use of concepts of one model in other models and guarantee the overall view of both the enterprise and the IS. Complete traceability is supported by the meta-model without the need to introduce additional explicit traceability links. The specific fine-grained inter-model relationships are further discussed in their relevant subsections below.

Figure 10. Overview of the Unifying Meta-Model: the coarse-grained relationships between the models.

The meta-model presented here follows a UML-like notation. Package symbols are used to denote models in Figure 10, implying that concepts
defined in the meta-model are spread over the six models. Furthermore, generalization links between concepts denote a general-specific relationship, whereas cardinalities on the relationships express the number of model component instances that can participate in a model relationship instance.

6.1. Common Components Model

All models described by the meta-model are derived from common components that provide the basis for other components. The Common Components Model (CCM) is shown in Figure 11. It is not included in Figure 10 because it is a conceptual abstraction that spans all other models. The *model component* is the topmost concept in the meta-model. Each model component has a unique identifier and a text field that allows the component to be labeled with a single name, a sentence, or a long text depending on the modeling needs. A modeling component has a *description*, which is a text that provides additional clarification for the component.

![Diagram of Common Components Model](image)

**Figure 11. The Common Components Model.**

The *relationship* concept is a model component that connects other model components with each other. Relationships represent first-class modeling entities in the unifying meta-model to emphasize their importance as modeling components and provide additional flexibility for linking other modeling components. Furthermore, explicating treating relationships as modeling concepts that are used to connect other modeling concepts facilitates additional control during the implementation of the unifying meta-model, as will be demonstrated in Chapter 8. Two distinct types of relationships are defined in the meta-model:

- **Intra-model relationships** link components within the same model. They are graphically represented as blue boxes with red contours; and
• *Inter-model relationships (IMRs)* enable components from different models to be related to each other. The inter-model relationships facilitate traceability among the models. They provide mechanisms to design intersecting models, whereby models share the use of common concepts. The IMRs are graphically represented as red boxes with red contours.

6.2. **Concepts Model**

Concepts that are necessary to describe the static aspects of enterprises and IS are modeled in the *Concepts Model (CM)*, shown in Figure 12. They include resources and information objects that are used, processed, exchanged, produced, and stored in the organization, together with their relationships and attributes.

![Figure 12. The Concepts Model.](image)

A *concept* represents things and phenomena about which the enterprise stores or processes information. Concepts represent resources, information objects, or other things that are of interest to the enterprise. They are described by *attributes* that declare properties of the concepts. Concepts can be related to each other using *concept relationships*, which are of the following kinds:

• *Binary relationship*: which is a general relationship that describes a connection between two concepts, where each instance of one concept can be related to instances of the other concept;

• *Generalization* relationship, which relates a general concept to a more specific one; and

• *Aggregation* relationship, which is used to indicate that a concept is composed of other concepts.
6.3. Goal Model

Organizational business goals are recorded and represented in the Goal Model (GM), as described in Figure 13. A business goal is a future state-of-affairs that the enterprise aims to attain, and through which it can grow and generate profit. An enterprise can identify potential desirable situations as opportunities, which highlight new possibilities or capabilities that can be transformed into actual business goals. Both opportunities and business goals are defined as types of intentional components because they share many properties and associations with other concepts, both in the GM and in other parts of the meta-model (e.g. both goal and opportunity are defined by a role). Moreover, modeling opportunities as intentional components allows the identification of related concepts, roles, processes, and requirements—otherwise associated only with business goals—to improve the contextual representation of opportunities and facilitate better judgment on whether the opportunity should be designated as a business goal. Intentional components can support each other, indicating that achieving one contributes to achieving the other. Intentional components can also conflict with each other when the realization of one component challenges the realization of the other. The role that defines an intentional component is captured in the meta-model to provide traceability to the source of the component.

![Figure 13. The Goal Model.](image)

The operationalization relationship provides additional structure to the GM. It allows business goals to be decomposed into smaller, more concrete sub-goals. Decomposition can occur in one of three types (or modes):
Operationalization enables the intentional components to be organized as a hierarchy. Goals have roles that are responsible for them; tracking the progress of the fulfillment of the goals and making sure necessary resources are allocated for that purpose.

Achievement of business goals is usually hindered by various obstacles. It pays to include those obstacles in the model to obtain a clearer view of the organizational landscape. Problems that hinder business goals can be either internal to the enterprise, in which case they are considered weaknesses, or external, in which case they are modeled as threats. The cause that explains a problem is a useful insight when identified explicitly, and can contribute to the identification of suitable measures and solutions. In addition, a business goal is bound by constraints, which represent rules and regulations that affect how the organization operates. Constraints are always external to the organization; internal rules and regulations are described using the Business Rules Model (see section 6.6).

### 6.4. Business Process Model

Business goals identified in the GM give rise to, or motivate, the design of business processes that describe activities in the enterprise needed to realize the goals. The Business Process Model (BPM) provides a view over the processes and their composition and structure (Figure 14). A process model component stands for different sizes of processes at both the business and IS levels, thus providing a dynamic view of the enterprise and its IS. The relationship between a process and its sub-processes is captured as a composition relationship, indicating that the sub-processes work together to accomplish the top process. The meta-model includes no limit to the number of decomposition levels, which is left to the specific needs of projects.
The sequencing of processes is described using the *process flow* relationship, which connects processes in one execution flow. The type of the relationship indicates whether the processes are: performed in parallel: using AND connection; optional: using OR connection; or conditional: using XOR connection. Processes are affected by *events*, which are external occurrences that influence the execution of the process and cause it to deviate into certain paths, e.g. at the decision points. Concepts that are *consumed* and *produced* are included in the BPM using inter-model relationships, denoting the inputs that guide the process execution and outputs that result from the execution.

Actors that *perform* processes in the enterprise are modeled as *roles*, which can also *provide* and be *responsible for* goals, and can be related to requirements. A role stands for a general position that is independent of the actual person filling it, and it can represent physical persons, virtual persons, or automated systems.

### 6.5. Requirements Model

High-level requirements, also called business requirements, express the stakeholders’ desires for a future IS. Business requirements are refined into more concrete IS requirements that are better understood by system designers. The line separating business requirements and system
requirements is vague and hard to identify, but the decomposition is always necessary. Therefore, the meta-model for the Requirements Model (RM) includes a single requirement component that serves as a high-level as well as a concrete system requirement (Figure 15). Decomposition of a requirement occurs on any level, and continues depending on the judgment of the modeler and the specific needs of the project. The decomposition relationship is used to connect a parent requirement with its child requirements, which can be all necessary (AND decomposition), alternatives (OR decomposition), or exclusive alternatives (XOR decomposition). Requirements that negatively affect the realization of each other are connected using the conflict relationship. Also, requirements that positively affect the realization of each other are connected using the dependency relationship.

Requirements at different levels of decomposition are motivated by business goals. However, some requirements address system related issues that are not relevant to the high-level organizational goals. When those requirements are elicited, information systems constraints that motivate or hinder them must be identified and assessed.

Requirements provide a central connection point between the models. During modeling, the actors, things, and activities that are involved in a requirement are identified and the respective roles, concepts, and processes are linked with the requirement. A requirement can have qualifiers that are used to express additional information about the requirement, e.g. performing the functionality at certain times or periodically, or the location for storing the data.

Figure 15. The Requirements Model.
6.6. Business Rules Model

Internal rules and regulations that govern the enterprise provide boundaries for the concepts and business processes. Business rules are often formulated together with the business goals to specify how the goals will be achieved. Concepts and business processes that are motivated by the business goals are governed by the defined business rules, and this is captured in the Business Rules Model (BRM), which is illustrated in Figure 16.

Business rules are represented in the meta-model as model components that are motivated by business goals. In other words, the business rules affect the fulfillment of business goals. A business rule refers to one or more concepts that are constrained by it. When a rule defines a necessity that needs to be guaranteed at all times by the involved concepts, it is called a structural business rule. When a rule addresses derived or dynamic properties that must be checked at certain points in time or when certain events occur, it is called a behavioral business rule. This type of business rules constrains the enterprise in terms of the change of its state, and can lead to different results depending on the triggering events.

Breaking a structural business rule produces an invalid state for the enterprise, and hence must be prevented. However, it is possible to break behavioral rules, and such breaches entail corrective actions that restore the enterprise to a valid state. Whereas structural rules constrain concepts, behavioral rules constrain processes, and can motivate the design of additional processes that are needed to enforce the rules.

![Figure 16. The Business Rules Model.](image)
6.7. IS Architecture Model

Creating a complete IS model involves describing the design architecture that the final system will be based on, and how its different parts will operate conjointly. It also involves describing how eventual users will interact with the system, and other technical details related to the MDD platform used for the implementation. This information is captured by the *Information System Architecture Model (ISAM)*, which is technology-dependent and can exist in various forms for the same GM, BRM, CM, BPM, and RM. The ISAM must support inter-model relationships between components of the other models and specific architectural components, depending on the selected implementation architecture. Those relationships highlight the motivation behind architectural design decisions, and provide additional information that can be exploited when the models are transformed into an executable system.

In essence, creating the ISAM involves using models to illustrate the transformations needed to generate an implementation. Since the implementation of the system includes complementary parts that play different roles in the execution environment (e.g. data storage, process execution, user interfaces, etc.), the ISAM is bound to be large and complex. The complexity however is tolerable because of the holistic view offered by the ISAM of the relationships between the different models that describe the system and the corresponding implementation components that realize them. Furthermore, multiple ISAMs can exist simultaneously, signifying that the IS can be implemented in multiple ways to support the enterprise that is described by the other models of the unifying meta-model.

To illustrate the use of the ISAM, an implementation based on Web services is presented. As an illustrative example, the proposed ISAM covers the BPM and CM. The details related to other models, such as user interfaces and business rules enforcement, provide the basis for future research.

To be able to describe how the modeling components of the BPM and CM will be implemented as Web services, we adapted the WSDL 2.0 meta-model published by WSDL (2007). The adapted meta-model is shown in Figure 17, together with the IMRs that associate the modeling components in the unifying meta-model with their counterparts in WSDL 2.0.
Aside from the standard components of WSDL 2.0, we introduce a modeling component to represent the description, service, interface, and interface operation of WSDL. The new WSDL process provides the necessary flexibility for the modeler to choose the granularity level at which the business processes defined in BPM will be implemented. A modeler can map a business process to the whole Web service description, to an interface, or even to a specific operation in an interface. The flexibility of breaking down processes to the suitable granularity level in BPM allows modelers to cover a wide range of processes of varying complexity and abstraction levels. This eliminates the need for additional intermediate models. The transition from BPM to the WSDL implementation is achieved using IMRs (the blue arrows in Figure 17). The concepts consumed and produced by a business process are consequently mapped to WSDL inputs and outputs. The type of the concept is transformed into a WSDL type and used as the type of the inputs and outputs. The types can be separately captured using XML schemas, which can facilitate the creation of data storage back-ends for the Web services to communicate with. This question however requires further investigation. Finally, the description modeling component, associated with the generic Modeling Component concept, is mapped to the documentation component in WSDL.

In addition to the mappings, the ISAM allows Web services to be modeled graphically. This explains including the whole WSDL 2.0 meta-model as part of the unifying meta-model. Modelers have the ability to specify implementation details not covered by other models, such as
bindings and end points, leading to a complete implementation description that can be readily used to generate the operational code of the Web services.

6.8. Summary

This chapter presented unifying meta-model for integrating EM and MDD. The meta-model provides the basis for a comprehensive approach for bridging the gap between EM and MDD. It offers multiple views of the enterprise and its supporting IS. The views cover the goals, business processes, business rules, requirements, concepts, and IS architecture of the enterprise and its supporting IS are described and interrelated. IMRs provide inherent traceability between the modeling components of the different views. Multiple implementation platforms are supported through designing a new ISAM for each platform. The ISAM presented in this chapter targets a Web service-based implementation. The unifying meta-model presented in this chapter is designed to support the integration properties discussed in Chapter 5. It is evaluated using an example case study and a prototype tool in Chapter 8.
7. Implementation of the Unifying Meta-Model

The increasing reliance on MDD for designing and implementing IS in recent years has motivated a large number of research projects, approaches, and tools that advocate the use of models as the main development artifacts. Models in MDD are used to capture various aspects of an IS, and (automatic) transformations enable the derivation of models from each other and generate executable code. In order to capitalize on the full potential of MDD principles, MDD approaches need to be supported with tools to facilitate the creation and management of models, meta-models, and transformations. Tools also enable the execution of model transformations and, eventually, code generation. Aside from that, tools should offer practical and usable functionalities that simplify the complexity associated with managing models and transformations (Henkel & Stirna, 2010).

The following sections answer the research question RQ4 by presenting an account of two tooling environments that are associated with MDD; namely, MetaEdit+\(^{11}\) and the modeling environment available in Eclipse\(^{12}\) through the Eclipse Modeling Framework (EMF)\(^{13}\) and the Graphical Modeling Project (GMP)\(^{14}\) plug-ins. The aim of this chapter is to explore the suitability of MetaEdit+ and EMF/GMP as environments for developing a supporting tool for the unifying meta-model. The environments themselves are considered MDD tools because they apply the principles of MDD. Although both differ from other traditional MDD tools in being used to develop tools, they are still amenable to the qualities of MDD tools discussed in the next section. The contents of this chapter are based on an earlier conference publication (see section 1.3).

7.1. Qualities of Model Driven Development Tools

The importance of MDD tools that is emphasized in the literature (see Chapter 4) has yet to lead to the creation of a tool that covers all the necessary aspects of MDD. Several of the works surveyed in Chapter 4 have addressed the requirements for MDD tools. They pointed out the drawbacks of existing MDD approaches in terms of tooling, and highlighted the lack of

---

\(^{11}\) http://www.metacase.com/

\(^{12}\) http://www.eclipse.org/

\(^{13}\) http://www.eclipse.org/modeling/emf/

\(^{14}\) http://www.eclipse.org/modeling/gmp/
tools that can realize the theoretical benefits of MDD. As indicated by Atkinson and Kühne (2005), many works that tackle MDD tools are limited to superficial comparisons of feature lists, with no deep insights into the selection of a tool based on realistic project requirements and available technical abilities and standards compliance. As a result, Atkinson and Kühne (2005) suggest a conceptual framework for MDD tool architectures that covers major existing MDD theoretical configurations. The proposed tool architecture captures a three-dimensional view of modeling levels by identifying three types of modeling abstractions:

- Form instantiation, where instances of one level follow the structure defined by types on a higher level of abstraction. It is manifested as classes and instances of the classes, which is the basis for object-oriented development. Form instantiation is also referred to as linguistic classification.

- Content instantiation, where instances include content that logically conforms to the content types described on a higher level of abstraction.

- Generalization, where elements specialize other elements on the same level of abstraction. Generalization is orthogonal to the other two abstraction types.

Levels spanning is a notion that complements the three dimensions. It indicates that a concept on a given abstraction level is related (along one of the mentioned dimensions) to instances on more than one other level. For instance, the meta-model level of the OMG four-layer modeling architecture spans the form of elements on the model and instance levels.

Using the architecture proposed by Atkinson and Kühne (2005), MDD tool developers can plan the number and types of modeling levels that will be supported by the tool. It is also possible to combine multiple levels in a single compact level. Examples of how the architecture is implemented in tools are given in the following sections.

Aside from the core MDD principles that tools must support (i.e. the creation and management of models and meta-models, and the creation and execution of transformations), the following quality areas are highlighted in the literature as necessary to enhance the process of creating and managing the models in MDD tools:
Understandability: this refers to the ease with which tool users are able to create meta-models and models (Pelechano, Albert, Muñoz, & Cetina, 2006). Understandability can be enhanced by tools through facilities that highlight and explain the intended purpose of parts of the models to tool users (Henkel & Stirna, 2010). The graphical notation, which is generally part of the definition of the modeling language, can be enhanced by the graphical editor in the tool (Pelechano et al., 2006). Furthermore, tools should support the representation of details on several levels of abstraction (Henkel & Stirna, 2010), which can occur along multiple axes (Atkinson & Kühne, 2005). A usability framework, including a conceptual model for capturing tool usability and an experimental process for conducting the usability evaluation, is proposed by Panach et al. (2011) to measure the satisfaction, efficiency, and effectiveness of an MDD tool.

Model evaluation: tools can offer support for model analysis and evaluation (Oldevik, Solberg, Haugen, & Møller-Pedersen, 2006). Evaluation is referred to as observability or “model-level debugging” (Uhl, 2008) when the reporting of warnings and errors is done during the creation of the model and the execution of the transformations—a similarity to what compilers do for programming languages.

Executability: the use of model-to-text transformations to derive (generate) executable program code from models that describe the desired IS (Henkel & Stirna, 2010; Oldevik et al., 2006; Pelechano et al., 2006).

Model repositories: tools must provide mechanisms for serializing models in order to transport them to other tools or store them for reuse (Henkel & Stirna, 2010; Oldevik et al., 2006; Pelechano et al., 2006). Serialization is another form of model-to-text transformation. However, it does not generate executable programming code. Storing models in repositories also raises the question of model integration (Henkel & Stirna, 2010).

Traceability and change management: MDD involves the use of transformations to advance from one stage of modeling to the next. Changes in earlier models need to be tracked and reflected in later
models, and tools must provide the necessary facilities to realize that (Oldevik et al., 2006).

- Other SE activities: since MDD tools are used in IS development projects, activities related to the development process itself are not isolated from MDD activities and must be supported by the tool. Project planning (Oldevik et al., 2006), collaborative development (Henkel & Stirna, 2010), and Quality of Service (QoS) management (Oldevik et al., 2006) are some of the activities highlighted in the literature.

- Tool documentation: finally, tools need to provide self-documentation—guidelines for using the tool itself.

The list of quality criteria described above is by no means complete. There could be other desirable qualities of MDD tools. However, the focus of this section is on criteria that are relevant for reporting the experience of using MetaEdit+ and the Eclipse MDD plug-ins to implement a supporting tool for the unifying meta-model.

### 7.2. Implementing the Tool in MetaEdit+

MetaEdit+ is a tool for developing Domain Specific Languages (DSLs), which are non-generic modeling languages that are designed for a specific application domain (Mernik, Heering, & Sloane, 2005). DSLs substitute the generic modeling capabilities offered by general-purpose languages (GPLs), such as UML, for more expressiveness that results from tailoring the language to the needs of a defined domain. Notations that are familiar to domain experts are utilized instead of generic shapes with broad semantics. Domain conventions and abstractions are also incorporated in DSLs. According to Mernik et al. (2005), DSLs are not required to be executable. However, when used in the context of MDD, a DSL needs to be transformed into other models and eventually into an executable form, following the MDD principles.

The modeling environment of MetaEdit+ is divided into two main parts: MetaEdit+ workbench and MetaEdit+ modeler. The workbench, shown in Figure 18, includes the necessary facilities for creating the meta-model of the language (in this case the unifying meta-model), organized as a set of tools. The whole meta-model is called a graph, and components of the meta-
model are created in the workbench using the object tool, property tool, and the relationship tool.

![The MetaEdit+ workbench.](image)

The object tool (Figure 19-a) is used to define concepts in the meta-model, where each concept is defined as an object that has a name, an ancestor (a super type), a description, and a list of properties created using the property tool (Figure 19-b). In turn, each property has a name and a data type, and it is possible to define the input method for creating property values and constraints on those values. For example, a property can have the type “string,” an editable list input widget to indicate that the property can have multiple values, and a regular expression that governs the values that can be entered by the user. Nesting of objects is made possible by allowing objects to be selected as the types of properties.

Relationships are defined in the workbench using the relationship tool in a similar manner to objects. In fact, relationships are treated in MetaEdit+ as individual modeling components that can have properties of their own. Roles are used to connect concepts to relationships following the principles of Object Role Modeling (ORM) (Halpin & Morgan, 2008), a conceptual modeling method that focuses on the separation of concepts, relationships, and the roles played by concepts in the relationships in which they participate. To realize that, the workbench includes a role tool that can be
used to create roles and assign properties to them. In general, and for the purposes of implementing the unifying meta-model, roles can be kept as simple connection points. However, the workbench allows more control over the definition of roles because they are also treated as standalone modeling components.

![Image of MetaEdit+ object tool and property tool](image)

Figure 19. The MetaEdit+ object tool (a) and property tool (b).

The graph tool (Figure 20) is where all modeling components are combined to create the meta-model. In the graph tool, concepts and relationships that are part of the meta-model are selected, and concepts are assigned to relationships using the defined roles. Additional management is also possible, such as creating constraints on the number of different relationships in which a concept can play a role.

7.2.1. MDD Tool Qualities in MetaEdit+

According to Atkinson and Kühne (2005), MetaEdit+ implements the two-level cascading tool architecture. Tool users design a model on an upper level using the built-in tool format in the workbench, thereby creating the meta-model of the DSL that is being developed. The tool uses the meta-model to generate another tool, the MetaEdit+ modeler, which is used to
create models on a lower level, representing instance models that follow the definition of the DSL. When the CM and BPM views of the unifying meta-model were implemented in MetaEdit+, the two views were created using form-based interfaces (some of which are shown in Figures 17 and 18) in the workbench. The resulting meta-model was maintained using a proprietary format. In other words, the meta-models were stored in a binary format that is only accessible in MetaEdit+. Then, the workbench generated a new modeler (Figure 21) that was used to create instance models.

![Image](image.png)

Figure 20. The MetaEdit+ graph tool.

When it comes to understandability, the form-based interfaces of the workbench tools hinder the ability of the tool user to gain an overview of the whole meta-model that is being implemented. Lists of objects, properties, relationships, and roles can be viewed separately, and only using the graph tool can they be viewed at the same time. Even then, the connections between objects and relationships using roles can only be seen individually, without a full overview. This limitation became a real obstacle during the development of the unifying meta-model because repetitive and short
change cycles meant that the tool user had to maintain a mental image of how the meta-model was being changed. Eventually, a separate copy of the meta-model was created using Microsoft Visio\textsuperscript{15}, spawning additional complexity for maintaining two versions of the meta-model.

Unlike the workbench, the MetaEdit+ modeler has a graphical interface for creating the models. The graphical representations of modeling components used in the modeler are defined in the workbench using a WYSIWYG tool. However, no additional support is offered by the tool in terms of explaining what each graphical symbol in the modeler means, limiting the understandability that can be offered during the creation of models.

Meta-models and models created using MetaEdit+ are both stored in a model repository that is part of the tool. The repository is a binary model database, the content of which can only be viewed using MetaEdit+. Model reuse is supported by the existence of the repository. However, transporting models requires additional user intervention and is not directly supported by the tool. MetaEdit+ includes a scripting language, called MERL, which can be used to traverse models and generate corresponding textual reports. Using MERL, tool users can create generator scripts that can be used to output any text based on the structure and content of the model, including serialization standards (e.g. XMI\textsuperscript{16}) or human readable reports (e.g. in HTML). During the implementation of the unifying meta-model, model executability was enabled using generators which translated BPM models into Java classes and CM models into XML schemas, but this required additional effort since the classes and schemas needed to be constructed from scratch using output statements in MERL.

The need for observability is eliminated by the tool architecture used in MetaEdit+, since models are not allowed to deviate from the way they are supposed to be constructed as dictated by the meta-model. For example, attempting to create a relationship between two instances of the wrong classes would prompt the tool to display an illegal operation message. While being an advantage when creating models, the inability to arbitrarily connect instances or assign attributes limited the ability to experiment during the development of the unifying meta-model. Testing new structures was not

\textsuperscript{15} http://office.microsoft.com/en-us/visio/
\textsuperscript{16} http://www.omg.org/spec/XMI/
possible. Atkinson and Kühne (2005) also stress this limitation when discussing the tool architecture.

Figure 21. The MetaEdit+ modeler for the CM view.

Among the various SE activities that can be supported by MDD tools, only collaborative development is available in MetaEdit+, realized using multiple user accounts that can simultaneously access the same model repository.

Extensive documentation exists for using different parts of MetaEdit+, including the tools in the workbench, the modeler, and the scripting language. Tutorials are available to guide beginners, and there is an active community that can provide help when needed. MetaEdit+ offers additional modeling functionalities that can be helpful for complex DSLs, such as model embedding, where a single modeling component can be further described using another model, as well as the ability to reuse modeling components across multiple models. However, these advantages are overshadowed by the inflexibility of the form-based interfaces and the two-level cascading architecture.
7.3. Implementing the Tool in Eclipse

Since its introduction as an Integrated Development Environment (IDE) over a decade ago, Eclipse has grown to be one of the major software development platforms available today. The open source and free software nature and the plug-in based architecture contribute to a wide and highly customizable range of Eclipse modules. A large community is involved in developing Eclipse plug-ins, and the development is organized in projects that focus on certain domains. The Eclipse Modeling Project\(^\text{17}\) focuses on modeling standards, frameworks, and tooling, and includes the Eclipse Modeling Framework (EMF) and the Graphical Modeling Project (GMP), which were used to implement the unifying meta-model and develop its graphical editor.

EMF provides facilities for building applications using models and model transformations. Meta-models can be created in EMF using Ecore, which is an implementation of Essential MOF (EMOF), a subset of MOF\(^\text{18}\) that is aligned with implementation technologies and XML. This highlights the mindset assumed by EMF as a component in an implementation of the Model Driven Architecture (MDA)\(^\text{19}\) (Steinberg, Budinsky, Paternostro, & Merks, 2009). Ecore meta-models can be acquired from models written in annotated Java code, XML, or UML. The meta-models are transformed into Java code that can be used to create, edit, and serialize models. EMF is also able to generate a basic editor for the models. However, creating a rich graphical editor for models is made possible through the functionalities of the GMP plug-ins (Gronback, 2009). The combination of EMF and GMP enables the development of DSLs and accompanying rich graphical editors in alignment with MDD principles.

The process for implementing the unifying meta-model using EMF and GMP is outlined in Figure 22, and is based on the recommended process in GMP. The first step of the process is to create the domain Ecore model, which in this thesis covers the CM and BPM views of the unifying meta-model. The domain model is acquired from a UML model designed in Papyrus\(^\text{20}\), which is an Eclipse plug-in for creating UML models. EMF includes a GMP-based graphical editor that can be used for directly creating

\(^{\text{17}}\)http://www.eclipse.org/modeling/
\(^{\text{18}}\)http://www.omg.org/mof/
\(^{\text{19}}\)http://www.omg.org/mda/
\(^{\text{20}}\)www.papyrusuml.org/
the domain model. However, it suffers from synchronization problems and changes are not always correctly reflected in the Ecore model. Papyrus represents a good alternative, especially since it offers a familiar modeling experience, itself being built using GMP.

The domain model is then used to create the generator model, which is a facility available in EMF to extend the domain model with implementation-specific details that are outside the scope of the meta-model. These details include features that can be enabled, disabled, or customized during code generation, such as interface naming patterns and operation reflection. The generator model is the one used in EMF to actually generate the Java classes for creating and editing models. An excerpt from the generator model is shown in Figure 23.

Two complementary models that are part of GMP are derived from the domain model. The graphical definition model describes the graphical notation that is used to create instances of the meta-model which is being implemented. The tooling definition model describes the palette that will be generated as part of the graphical editor to select and create modeling components. By using these models, GMP separates the meta-model from its visual representation and from the functionality used to create it. The graphical definition model and the tooling definition model can be edited separately from the domain model to customize the graphical editor. For instance, Figure 24 shows the graphical notation of the concept modeling component of CM, defined as a rounded rectangle that includes labels for...
the id, name, and description properties of the concept. Similarly, the graphical representations of other concepts and relationships can be defined. GMP offers a range of basic shapes that can be customized and embedded within each other, contributing to a flexible notation definition tool.

Figure 23. The generator model of EMF.

A mapping model is created to integrate the domain model, graphical definition model, and tooling definition model. The integrated model is finally used to derive an editor generator model that adds the necessary implementation-specific details, and has a role similar to that of the generator model in EMF. Figure 25 illustrates the mapping model. It shows how a node mapping for the concept modeling component associates the concept definition (in the domain model) with its graphical notation (in the graphical definition model) and its tool (in the palette created from the tooling definition model).
Figure 24. The graphical definition model of GMP.

The code generated from the editor generator model is combined with the code generated from the EMF generator model to create the graphical editor. GMP can generate the editor as a standalone Rich Client Platform (RCP) that is solely used to create and edit models. It can also generate an Eclipse plug-in that can be installed in an Eclipse environment and used in combination with other plug-ins.

7.3.1 MDD Tool Qualities in EMF/GMP

Positioning the combination of EMF and GMP within the tool architecture proposed by Atkinson and Kühlne (2005) reveals how they constitute an MDD tool that realizes level compaction while providing spanning at the same time. Level compaction refers to the use of one representation format to capture two modeling levels (Atkinson & Kühlne, 2005), and this is achieved in EMF by using Ecore to describe the structure of domain and instance models. As mentioned earlier, Ecore is an implementation of a subset of MOF called EMOF. Since MOF represents the format abstraction of meta-models and models, the models can serve as meta-models that can be further instantiated. In other words, the graphical editor generated for the
unifying meta-model can be used to design models that can serve as domain models for creating new graphical editors, and this cycle can be repeated endlessly. As for spanning, the use of MOF and the model-editing framework provided in EMF enable the creation of different modeling languages, be they DSLs or GPLs, using the same generic mechanisms. A desired side effect is the increased understandability by tool users; developing a new DSL does not require additional knowledge of the tool. Using a standardized language for meta-modeling, i.e. MOF, further enhances understandability because the semantics of the language are well defined.

Figure 25. The mapping model of GMP.

Another factor affecting understandability is the ability to acquire domain models from a variety of sources. Tool users who are familiar with Java code can use annotated Java to design the meta-model. Users who prefer XML or UML can continue to work in their usual ways, and EMF will take care of deriving the Ecore domain model from the input meta-
model. The existence of a plethora of UML graphical modeling tools, both as standalone environments and as Eclipse plug-ins, allows further flexibility in creating the domain model. In fact, the switch from the EMF built-in Ecore graphical editor to Papyrus was seamless, made easier by the fact that both editors are built using GMP and offer similar user experiences.

Both EMF and GMP include a general Eclipse plug-in for error highlighting and management. Using the wrong value for a property or not entering the value of a required property when editing EMF and GMP models prompts Eclipse to highlight the involved property and display an explanatory message. Suggested solutions can also be provided for common and recurring types of problems.

The models that are created in EMF and GMP are encoded in XML and eventually transformed into Java code. They are human-readable as well as transportable to other tools or platforms. The open source nature of Eclipse enables tool users to access the source code of the plug-ins and customize their functionality if necessary, changing the generated code to suit their needs. The Eclipse plug-ins also support model integration, which occurs at two points in the process illustrated in Figure 22:

- The mapping model integrates the domain, graphical definition, and tooling definition models; and
- The final graphical editor is created by integrating code generated from the editor generator model with code generated from the EMF generator model.

Wizards also exist to guide users through repetitive and common activities.

Traceability is partially managed by the facilities of EMF and GMP through the use of naming conventions. Consequently, model parts and generated code can be traced back to their sources. However, traceability is limited in parts of GMP due to the separation of the domain model from its graphical notation and tooling. Changes to the graphical definition and tooling definition models will be lost if the models are regenerated following changes to the domain model.

The large community behind the development of Eclipse guarantees a wide range of plug-ins that can support the software development process. Code repositories, task lists, collaboration, project planning, and integration with other tools are only a few of the domains for which plug-ins can be
found. Furthermore, resources associated with Eclipse in general and with EMF in particular are plentiful, both as books and on the Internet. However, up-to-date resources on GMP are scarce and hard to find.

7.4. **Summary**

The observations reveal that both tools have advantages and drawbacks. On the one hand, the separate tools available in MetaEdit+ for creating and managing concepts, properties, relationships, and ports, in addition to the graph itself (i.e. the meta-model), provide increased control over the definition of the meta-model. MetaEdit+ combines all modeling components in a single graph (albeit using multiple user interfaces). A single transition was then required to generate an editor, and only one modeling technique was involved. But these advantages were limited by the use of a proprietary modeling technique and the lack of a single and complete view of the whole meta-model. The need for the overall view dictated the reliance on an external tool (Microsoft Visio), multiplying the effort needed to maintain the meta-model during its development and evolution.

On the other hand, EMF and GMP implemented well-known and open standards. The visual representation of the domain model in EMF offered the necessary overall view and shortened the cycle of updating the meta-model and testing the changes for their suitability. EMF and GMP used many models and automatic transformations between the models, constituting an MDD approach that covered several layers of modeling. This separation is necessary to decouple unrelated information, but it resulted in a long development process that involved many types of models. Consequently, different types of modeling knowledge were required. Table 5 summarizes the extent to which MetaEdit+ and EMF/GMP fulfill the MDD tool qualities identified in section 7.1.
Table 5. Summary of MDD tool qualities as perceived in MetaEdit+ and EMF/GMP.

<table>
<thead>
<tr>
<th></th>
<th>MetaEdit+</th>
<th>Eclipse plug-ins of EMF and GMP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Understandability</strong></td>
<td>+ Consistent interfaces of all tools.</td>
<td>+ Uniform usage of models and meta-models because MOF is the meta-meta-modeling language.</td>
</tr>
<tr>
<td></td>
<td>+ Graphically create models.</td>
<td>+ Support for many sources and formats to acquire domain models, hence adaptable to user skills.</td>
</tr>
<tr>
<td></td>
<td>– No overview of the whole meta-model.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>– No explanation during the creation of instance models.</td>
<td></td>
</tr>
<tr>
<td><strong>Model Evaluation</strong></td>
<td>+ Tool architecture eliminates the need for evaluation: not possible to create models that do not conform to the meta-model.</td>
<td>+ Ability to use a general Eclipse plug-in that provides error highlighting and suggested solutions during development time.</td>
</tr>
<tr>
<td><strong>Executability</strong></td>
<td>+ Scripting language for generating any text from models.</td>
<td>+ Many Eclipse plug-ins to support executability (e.g. Java Emitter Templates JET).</td>
</tr>
<tr>
<td></td>
<td>– No generation framework (e.g. for Java code).</td>
<td></td>
</tr>
<tr>
<td><strong>Model Repositories</strong></td>
<td>+ Model reuse.</td>
<td>+ XML-based: human-readable and accessible by other tools.</td>
</tr>
<tr>
<td></td>
<td>– Proprietary binary storage format.</td>
<td>+ Java-based, enabling access and editing of code.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ Many Eclipse plug-ins to support model integration.</td>
</tr>
<tr>
<td><strong>Traceability and Change Management</strong></td>
<td>Not applicable because only one model is used.</td>
<td>+ Partially supported using naming conventions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– Limited in parts of GMP, causing changes to be lost if some models are regenerated.</td>
</tr>
<tr>
<td><strong>Other SE Activities</strong></td>
<td>+ Collaborative development.</td>
<td>+ Many Eclipse plug-ins to support a myriad of activities.</td>
</tr>
<tr>
<td><strong>Tool Documentation</strong></td>
<td>+ Extensive documentation.</td>
<td>+ Extensive documentation.</td>
</tr>
<tr>
<td></td>
<td>+ Large support community.</td>
<td>+ Large support community.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– No up-to-date resources on GMP.</td>
</tr>
</tbody>
</table>
8. Validation of the Unifying Meta-Model

This chapter continues addressing the research question RQ4. The application of the unifying meta-model and its supporting tool is illustrated using an example case study. The unifying meta-model is continuously being developed. The example case presented here serves as a proof-of-concept to demonstrate the value of the utilization of the meta-model and the supporting tool. The advantages of using the unifying meta-model for integrating the domains of EM and MDD are demonstrated, and the potential for improving the meta-model and the supporting tool highlighted. The extent to which the integration properties elicited in Chapter 5 are fulfilled is also discussed, alongside the future development plans to cover all the integration properties, leading to a comprehensive and streamlined IS development approach that applies MDD principles while covering the development path starting with enterprise models and ending with different implementation platforms.

8.1. Overview of the Example Case

Reach More Inc. is an imaginary marketing company with clients in many production and service domains. With the increasing influence of social media, Reach More Inc. wants to attain a competitive edge by developing new marketing strategies based on the trends found in major social media outlets. In order to achieve that, the company needs to uncover the relevant information that is being shared and circulated about a certain product. Reach More Inc. decides to use the unifying meta-model and the supporting tool to analyze its needs concerning the discovery of relevant information and trends found in social media outlets. The unifying meta-model will help develop an IS that collects and aggregates information from various social media outlets based on given criteria. The aggregated view is designed to help marketing consultants at Reach More Inc. to uncover marketing possibilities and formulate more effective and cutting-edge marketing strategies.

The unifying meta-model enables Reach More Inc. to formulate goals and opportunities and uncover possible problems and constraints. It will also enable the company to elicit the relevant concepts and design processes to acquire and aggregate the information accordingly from various social
media outlets using a SOA-based IS implementation that utilizes Web services for collecting and aggregating the information.

8.2. Tool Implementation

Based on the findings in Chapter 7, the Eclipse environment (including the EMF and GMP plugins) was selected to implement the supporting tool for the unifying meta-model.

The meta-model was initially developed in Microsoft Visio. It was then imported into EMF to create the corresponding Ecore files. The packages of the unifying meta-model, representing the different models, needed to be flattened before they could be used to develop the tool, since the Ecore implementation in EMF is not able to handle multiple packages. Consequently, the meta-model implementation consisted of a single package that included all modeling components from the different models. After that, the process illustrated in Figure 22 was followed to generate the intermediate models necessary for developing the tool. This included:

- Using the functionality provided in EMF to generate the Java code necessary for holding, accessing, and editing instances of the modeling components of the unifying meta-model;

- Deriving and updating the graphical definition model (gmfgraph), the tooling definition model (gmftool), and the mapping model (gmfmap). The editor generator model (gmfgen) was then derived from gmfmap; and

- Using gmfgen to generate the Java code for the graphical editor (i.e. the tool) for the unifying meta-model.

The implementation of this tool for the unifying meta-model is limited to some parts of the meta-model, namely the GM, CM, BPM, and a Web service-based ISAM. The implementation is intended as a proof-of-concept to illustrate the viability of using the unifying meta-model to realize the integration properties elicited in Chapter 5. The remaining parts of the unifying meta-model are still under development. Plans for future research and development are discussed in the following chapter.
8.3. Modeling Business Goals

In this example case, the entry point for modeling is the business goals (i.e. intentions) of Reach More Inc., since they govern the processes that will be designed and the concepts that the processes will work with. The topmost business goal is to “develop marketing strategies based on the trends found on major social media outlets,” captured as Business Goal 1 in Figure 26. The operationalization of this goals leads to creating the other goals in the figure. The two direct sub-goals are to “increase awareness of prevalent trends” (Business Goal 5) and to “rely on informed decisions to develop marketing strategies (Business Goal 6). Business Goal 5 is operationalized further into four additional business goals. In this example, all sub-goals are necessary for the fulfillment of their parent goals, which means that the AND variant of the operationalization relationship is used.

Modeling the roles that are responsible for each business goal indicates who is going to track the progress of the fulfillment of the goals and allocate the necessary resources for that purpose. Using the role “system developer” is a way to highlight the goals that will lead to the development of supporting IS. At the moment, the unifying meta-model is not capable of automatically detecting which goals require supporting IS. However, future versions will cover this capability with the help of role attributes and the relationships between roles and business goals.

8.4. Modeling Processes and Concepts

The business goals that were identified in the previous section give rise to new processes and related concepts (i.e. concepts that are consumed and produced by the processes). Figure 27 illustrates how Business Goal 3 motivates the creation of a process to design the aggregation model necessary to combine information from various social media sources (Process 3). This process consumes information about marketable objects (i.e. the objects for which marketing campaigns are created, including products and services) as well as the structure of the information found on social media outlets concerning those objects (represented by Concept 7). The output of Process 3 is an aggregation model that describes how the information about marketable objects will be combined, aggregated, and summarized. In this example case, the details of Process 3 are not revealed. They can be described on a finer level of granularity to identify which tasks
are required to complete the process. Both the system developer and the marketing consultant are involved in performing the process of developing the aggregation model. Furthermore, Concept 7 and Concept 10 in Figure 27 are illustrative, and their structures need to be elaborated, leading to a more complex design of concepts and their relationships.
Figure 26. Goal model of Reach More Inc.
Figure 27. The process and associated concepts derived from Business Goal 3.
Business Goal 4 motivates the design of a process for acquiring, aggregating, and displaying the information about a given group of marketable objects. The process is illustrated in Figure 28, and the relationship between Business Goal 4 and the constituent processes is illustrated by the “motivate” relationships. The creation of the new process with its sub-processes required the introduction of a new role, the “integration solution,” which represents the IS that will implement the aggregation functionality. Inputs and outputs of the sub-processes are captured as concepts, some of which have already been elicited together with Process 3 (see Figure 27). The final stage of the new process, namely Process 6, will not be part of the integration solution. This is indicated by choosing the marketing consultant as the role that performs Process 6. Process 6 can be elaborated further, and this might entail that additional IS will be developed to support the marketing consultant in creating the new marketing strategy. As with the previous figure, some concepts are illustrative, and their actual structures have more complexity.

The concepts model that has been created to complement the business process model requires a few additional concepts to cover all the information that Reach More Inc. finds necessary and relevant. The concepts model that represents Reach More Inc. is illustrated in Figure 29.
Figure 28. The process and associated concepts derived from Business Goal 4.
Figure 29. Concepts model for Reach More Inc.
8.5. Generating a Web Service-based Information System Skeleton

According to the ISAM that was presented in section 6.7, it is possible to realize processes, in WSDL 2.0 terms, as Web service descriptions (i.e. whole WSDL documents), Web services, interfaces, or operations. This is possible because the Web service meta-model was adapted to the unifying meta-model by extending it with a WSDL:Process modeling component which covers the four mentioned Web service components (see Figure 17). Following the ISAM, Reach More Inc. choose to implement Process 1 and Process 2 as two interfaces of Web services for handling all operations concerning social media APIs (Figure 30). In the same model, Process 4 is designated as a separate Web service for processing the retrieved information and producing the aggregated marketable object information. The flexibility offered by the BPM and ISAM enables Reach More Inc. to design nested Web service implementations to capture different levels of granularity for different business processes. Similarly, concepts are mapped to WSDL elements that are used as inputs and outputs in WSDL operations. Thus, the concepts are implemented using XML Schemas.

Since relationships in the unifying meta-model are treated as first-level modeling components, the ISAM is able to map relationships to Web service components. “Produce” and “consume” relationships, which connect processes to the concepts on which they operate, are mapped to input and output components in the Web service implementation. Combined with the mappings between concepts and WSDL elements, the inputs and outputs are able to correctly envelop their relevant elements.

The mappings to a Web service-based implementation can be used to design transformations that will realize an IS able to fulfill the goals of Reach More Inc. in a traceable and demonstrable manner. The meta-model is capable of expressing the transformations using IMRs that connect components of the ISAM to components in other models of the meta-model. More specifically, the IMRs comprise transformation rules that describe how an implementation can be graphically designed and automatically generated. Using such a model enables the tool to generate a skeleton of a SOA-based IS implementation consisting of Web service definitions (using WSDL) and XML Schemas. The concepts are translated into XML Schemas, which can be either kept in that form or translated into a relational database schema. Currently, the tool relies on Java Emitter Templates.
(JETs) to implement the transformation and generation mechanism. However, future plans include developing Query/View/Transformation (QVT) statements to provide more flexibility in describing and executing transformations. Further research is required to decide whether QVT will be chosen over other possible alternatives, and plans are currently made to carry out these steps.

Figure 30. Mapping processes and concepts to components of a Web service ISAM.
The process for generating the skeleton of a SOA-based implementation is divided into two phases: the tool design phase, and the tool runtime phase. These phases are described below.

The tool design phase covers the steps required for implementing the code generation functionality during the creation of the tool in Eclipse. It includes the following steps:

1. JETs are created to describe how the modeling components that are defined in the ISAM will be translated to code. Two JETs are currently created. The code snippet in Figure 31 illustrates a part of the JET used to generate WSDL documents based on the ISAM in the tool. The other JET, part of which is shown in Figure 32, is used to generate the XML Schema (also based on the ISAM).

2. The JET plug-in in Eclipse automatically translates every JET into a Java class that can be used for code generation. The class includes a “generate” method that accepts an arbitrary input parameter—usually a list of objects for translation—and returns a string that represents the outcome of the translation according to the instructions found in the JET. Special directives in the JET dictate the name and location of the Java class.

3. The Eclipse command framework is used to add a “generate” option to the popup menu associated with the tool editor view. A generator command handler is implemented in Eclipse to prepare the modeling components of the ISAM for transformation before passing them to the Java classes obtained from the JETs. The IMRs that connect the modeling components of the CM and BPM to their counterparts in the ISAM (the blue dashed arrows in Figure 30) are traversed to identify the input parameters to the JETs. Two input parameters are assembled: the first describes the Web services, their interfaces, and their default bindings, and the second describes the XML Schema definitions for the concepts, their attributes, and their containment relationships.
Figure 31. JET used for generating WSDL documents.
Figure 32. JET used for generating XML Schemas.
The tool runtime phase covers the steps performed while using the tool to generate the code according to the design of the ISAM. During this phase, two steps are performed:

1. The functionality implemented in the generator command handler is called. The ISAM of Reach More Inc. (shown in Figure 30) is traversed, and the input parameters described in step 3 of the tool design phase are prepared.

2. The assembled input parameters are passed on to the “generator” methods of the Java classes derived from the JETs. Applying the code of the JET described in Figure 31 generates, in the case of Reach More Inc., the WSDL document illustrated in the code snippet of Figure 33. This WSDL code can be used as it is, or it can be customized before being deployed and used in an operational environment. The code in Figure 33 demonstrates a generated Web service description where the operations of the interfaces refer to XML elements defined using an XML Schema. Applying the JET described in Figure 32 will generate the XML Schema, a part of which is shown in Figure 34.
Figure 33. Part of the generated WSDL document for Reach More Inc.
Figure 34. Part of the generated XML Schema for Reach More Inc.
8.6. Fulfilling the Integration Properties

The integration properties that were elicited and discussed in Chapter 5 serve as a list of requirements that are necessary for achieving integration between EM and MDD. The unifying meta-model and its supporting tool are designed to support the fulfillment of the integration properties. Table 6 follows the same format as Table 4 to summarize how the currently supported integration properties are realized and how the remaining ones are envisioned for fulfillment in the future.

Table 6. How the unifying meta-model fulfills the integration properties.

<table>
<thead>
<tr>
<th>Static and dynamic aspects</th>
<th>Intentional aspects</th>
<th>Model design guidelines</th>
<th>Model completeness</th>
<th>Architectural aspects</th>
<th>Change propagation</th>
<th>Model reusability</th>
<th>Tool support</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Unifying Meta-Model</td>
<td>Yes</td>
<td>Planned for the future</td>
<td>Partial</td>
<td>Yes (IMRs)</td>
<td></td>
<td>Partial</td>
<td>Prototype tool supporting many of the integration properties and offering additional support and functionality</td>
</tr>
</tbody>
</table>

The unifying meta-model is being actively developed and improved, and complete fulfillment of the integration properties is discussed in Chapter 9 as part of the future work. In its current version, the unifying meta-model addresses the following integration properties (which were first described in Chapter 5):

- Support requirements that define both static and dynamic aspects of the information system—the models that constitute the unifying meta-model are able to capture static and dynamic aspects of the enterprise and its supporting IS with the help of the CM and BPM, respectively. The CM is capable of capturing the concepts which are of interest to the enterprise and which are processed in the IS. At the same time, processes (i.e. activities) that use and manipulate the concepts are represented in the BPM. The ability to sequence those processes and divide them into smaller processes gives the modeler flexibility in defining activities on many levels of granularity, spanning the domains of EM and MDD at the same time.
• **Support intentional aspects of the requirements**—the implicit intentions for designing the processes of an enterprise and its supporting IS are exposed and captured with the help of the GM. Understanding the motivation for having certain processes will improve the process design and help alleviate potential problems and conflicts that would otherwise go undiscovered.

• **Support requirements that define architectural aspects of the information system**—architectural aspects are reflected in the division of the unifying meta-model into sub-models that address different views of the enterprise. Multiple views in the supporting tool will enhance the ability to view the enterprise from different architectural perspectives, and are planned for future versions of the tool implementation. However, if this capability is built into the meta-model itself, as is currently the case, then its realization in the tool will be easier (see the discussion in section 7.1 on MDD tool architectures for more on the advantages of supporting architectural aspects on the meta-model level). The ISAM offers the ability to implement the supporting IS in a variety of implementation software architectures. Nevertheless, a new ISAM must be designed for every new implementation architecture. In the case of Reach More Inc., a Web service-based implementation was chosen, reflecting the basic principles of SOA.

• **Enable change propagation between enterprise models and MDD models**—change propagation is achieved in the unifying meta-model by using IMRs, where changes in one model require the modeler to update the related modeling components in other models. No extra effort is required to achieve that, since IMRs inherently connect different models with each other. The tool can offer additional support for change propagation by highlighting broken IMRs, and this is part of the future plans for developing the tool.

• **Provide supporting tools**—the models presented in the example case were created using a tool implemented in Eclipse with the help of the EMF and GMP plugins. The tool currently offers basic support for creating and editing the models. However, future plans include extending the tool with multiple views to facilitate and simplify the creation of the models and to offer the relevant information to different types of users. For instance, the marketing consultant in
Reach More Inc. may not be interested in the technical details of the Web service implementation. These details become a burden if they are visible all the time. Active error reporting that points the modeler to missing information or relationships will contribute to improved model quality and completeness and is facilitated by the IMRs.

The way in which the unifying meta-model and the supporting tool fulfill the integration properties is open for improvement. In the next chapter, the potential for improving the currently supported integration properties is discussed, in addition to the plans for extending the coverage of the unifying meta-model and its supporting tool to all integration properties.

8.7. Summary

The purpose of this chapter is to demonstrate the utility and validity of the unifying meta-model for bridge the gap between EM and MDD. The fictional case of Reach More Inc. is used to illustrate how the meta-model can be instantiated, and how a skeleton of an IS implementation can be generated with the help of the different models that comprise the unifying meta-model. A prototype tool is developed to support the creation of instances of the unifying meta-model. The tool is used to carry out the example use case. Finally, the use of the unifying meta-model is evaluated using the set of integration properties presented in Chapter 5. The fulfillment of each integration property is examined, and the potential for improved support in the future is discussed.
9. Conclusion

Creating software by auto-magically pushing a button is the ultimate dream of researchers in IS and software development. Although it is a dream unlikely to be realized, it has driven researchers to find ways to streamline the development process and raise the abstraction level of the development artifacts. New methods and techniques are constantly emerging to increase the efficiency of creating IS and improving the alignment between the intentions for developing the IS and the functionality offered by it.

This thesis presented the unifying meta-model for enterprise modeling and model driven development, a modeling approach that provides a unified platform for designing enterprise models and transforming them into IS that are aligned with the intentions of the enterprise. The main research question of the thesis is addressed by the current design of the meta-model and its supporting tool. The example case presented in the Chapter 8 demonstrates how the unifying meta-model can be utilized to design and implement an IS that supports the goals of an enterprise. Intermediate research questions that were discussed in section 1.2 were addressed in the other chapters of the thesis.

Research questions were elicited at the beginning of the thesis, in section 1.2, to define the problems that this research tackles. RQ1 was designed to understand the current relationship between MDD and EM. To answer this question, an SLR was conducted, and the results were presented in Chapter 4. RQ2 aimed at creating a framework for integrating EM and MDD based on the answer to RQ1. The properties for integrating EM and MDD were presented in Chapter 5, divided into design-, model-, method-, and technology-based properties. Then, to address RQ3, which speculated on the components and characteristics of a modeling and development approach that fulfills the elicited integration properties, the unifying meta-model was developed (Chapter 6), and a tool was designed and implemented to facilitate the creation and management of instance models. Finally, the viability of the developed meta-model and the supporting tool, covered by RQ4, were demonstrated through an example case in Chapter 8.

Albeit not presenting a complete development process that fulfills all the integration properties presented in Chapter 5, the unifying meta-model is a step towards realizing a comprehensive and streamlined IS development process that can help design and implement IS that are derived directly from
the goals of an enterprise, while at the same time offering the flexibility of starting the modeling at any point of the development process and implementing the IS on any desired platform.

The development of the unifying meta-model took into consideration the current state-of-the-art of the relationship between MDD and previous stages in IS development, including RE and EM. A set of necessary integration properties of EM and MDD was compiled. The properties represent guidelines for researchers who are trying to integrate EM and MDD. They cover design-, method-, model-, and technology-related aspects. The unifying meta-model was designed with the integration properties in mind. It currently supports a subset of the properties, and plans exist for developing the unifying meta-model further. The ultimate goal is fulfilling all the properties, thereby offering a complete, comprehensive, and effective enterprise modeling and IS development approach. The applicability of the unifying meta-model was demonstrated with an example case study to illustrate its viability and highlight strengths and future development possibilities.

9.1. Future Work

The results presented in this thesis represent a first step in an ongoing research process. Further development and improvements of the unifying meta-model include:

- Enhancing the definitions of the models that were not covered in this thesis, and implementing the mappings between the ISAM and the other models using a transformation language. Query/View/Transformation (QVT) is a possible candidate for describing the transformations, but additional investigation is required for an informed choice.

- Extending the meta-model definition to capture other perspectives of the enterprise and its supporting IS and enhance the coverage of the approach. Potential new models include an organizational model, a user interface model, and a value model.

- Developing a method for designing the models. This includes criteria and guidelines for starting the modeling process with a certain model and how other models can be derived from it. It also
includes criteria for deciding which parts of the enterprise design will be supported through an IS.

- Investigating and enhancing the understandability of the modeling approach. The notation is currently generic and attempts to emulate some of the widely used notations in the software and IS modeling worlds. Moreover, model-level debugging is highlighted as a necessary integration property, and the implementation platform for the tool is capable of supporting such functionality.

- Improving the creation and management of models by business users through multiple views in the supporting tool. Relevant details can be revealed or hidden according to different user groups, simplifying the usage of the unifying meta-model.

- Packaging the unifying meta-model as a service on the cloud to improve accessibility and collaborative modeling capabilities.

- Validating the unifying meta-model and its supporting tool through real case studies. Testing an approach in controlled lab environment is helpful, but real-world problems can provide valuable feedback for improving the applicability of the approach and enhancing its usability.

The unifying meta-model requires further development to become a complete and practically applicable enterprise modeling and IS development approach. Many other extensions can also emerge along the way. This thesis is the first step towards creating a streamlined, flexible, and easy to use modeling approach that covers the path that starts with the conception of ideas through to the implementation of IS in different platforms.
References


