Ghent University

Faculty of Economics and Business Administration

Academic Year 2012–2013

Development of a Tool for Business Architecture Modeling in Eclipse

Thesis presented to obtain the degree of Master of Science in Applied Economic Sciences: Business Engineering

Simon Zutterman
Under supervision of Prof.dr.Geert Poels & Maxime Bernaert
Development of a Tool for Business Architecture Modeling in Eclipse

Thesis presented to obtain the degree of
Master of Science in
Applied Economic Sciences: Business Engineering

Simon Zutterman
Under supervision of
Prof.dr. Geert Poels & Maxime Bernaert
Permission

'The author gives the authorization to make this thesis available for consultation and to copy parts of it for personal use. Any other use is subjected to the limitations of copyright, in particular with regard to the obligation to explicitly mention the source when quoting results from this thesis.'

Simon Zutterman, May 2013
Preface

This thesis project has brought me both joy and sadness as there were happy times and difficult times. Thankfully, I never stood alone and had great support to push me over the line.

The document that lies before you is the result of hard work, and could not be made possible without the valuable contributions of a few people that I would like to thank.

First and foremost I want to thank my parents, for who I am extremely grateful for the support and confidence they give me each day. The fact that they gave me the opportunity to pursue a master’s degree is something many people would envy and something that I don’t just take for granted. Therefore I say to them: Thank you!

Next, I would like to thank Maxime Bernaert. He was my supervisor throughout the course of this thesis, made sure I did not stray off the correct path and he has always provided me with useful feedback and insights.

Last and surely not least, I want to acknowledge my great girlfriend and all my friends, who have supported me in difficult times, but were also around when the need for some relaxation presented itself!
Samenvatting

Deze thesis behandelt het ontwikkelen en valideren van een grafische tool steunend op het Eclipse platform. In tegenstelling tot een programmatische aanpak werd gekozen voor een model-gedreven aanpak, gebruik makend van het GMF framework.

De grafische tool wordt gebaseerd op het CHOOSE metamodel, die een nieuwe kijk op Enterprise Architectuur biedt. Door de nadruk op de Business te leggen en een beperkt aantal elementen te handhaven richt CHOOSE zich naar kleine-en middelgrote bedrijven. Na een korte studie omtrent Enterprise architectuur, specifiek gelinkt aan KMO’s, behandelt deze thesis het ontwerp en toepassen van de tool.

Na de ontwikkeling werd met behulp van case studies bij twee Belgische KMO’s de tool, en bijhorend de CHOOSE approach, getest en gevalideerd. Uit de onderzoeksfase werd duidelijk dat de tool een goede basis vormt om de CHOOSE methode toe te passen. Echter, naar de toekomst toe dienen bijkomende functies gerealiseerd te worden om de mogelijkheden die CHOOSE biedt ten volle te kunnen benutten.
Abstract

This thesis project concerns the development and validation of an Eclipse-based graphical editor. A model-driven approach was taken, making full use of the GMF Framework.

The editor takes inspiration from the CHOOSE metamodel, which offers a new view on Enterprise Architecture. By focusing on the Business layer, as well as maintaining a restricted amount of artifacts, CHOOSE targets small and medium-sized enterprises.

Following a short study on Enterprise architecture linked to SMEs, the author covers the design and practical use of the tool. After the development, the tool, along with the CHOOSE method, were tested and validated at two Belgian SMEs. It was concluded that the editor forms a good base for CHOOSE modeling, but requires further extensions in order to realize the possibilities that the CHOOSE method can offer.
## Contents

Preface i  
Samenvatting ii  
Abstract iii  
List of Abbreviations viii  
List of Figures x  
List of Tables xii

1 Thesis Overview 1  
1.1 Introduction 1  
1.2 Tool Support Development 2  
1.3 Evaluation and Conclusion 2

I Introduction 3

2 Bringing Enterprise Architecture to SMEs 4  
2.1 Small and Medium-Sized Enterprises 5  
2.1.1 What are Small and Medium-Sized Enterprises? 5  
2.1.2 Information Technology Adoption in SMEs 5  
2.1.3 SME Challenges 6  
2.2 Enterprise Architecture 8  
2.2.1 What is Enterprise Architecture all about? 8  
2.2.2 Existing Frameworks 9  
2.2.3 A Common Structure 12
3 CHOOSE Approach

3.1 Metamodel

3.1.1 Goal Viewpoint

3.1.2 Actor Viewpoint

3.1.3 Operation Viewpoint

3.1.4 Object Viewpoint

3.2 CHOOSE Integrated View Model

II Tool Support Development

4 Tool Support

4.1 Use of Tool Support

4.2 Software Quality

4.3 Requirements Analysis

5 State of the Art

5.1 The Eclipse Platform

5.2 The Eclipse Modeling Project

5.3 Eclipse Modeling Framework

5.3.1 Model Driven Architecture

5.3.2 Models, Metamodels and Meta-metamodels

5.3.3 Ecore

5.3.4 EMF Generator model

5.4 Graphical Modeling Framework

5.4.1 Domain Model

5.4.2 Tooling Definition Model

5.4.3 Graphical Definition Model

5.4.4 Mapping Model

5.4.5 Generator Model

5.5 Why the Eclipse Platform and GMF?

5.5.1 Open Source

5.5.2 Development Time and Editor Quality

5.5.3 Academic Praise and Support

5.5.4 Transformation Capabilities
List of Abbreviations

ATL ATLAS Transformation Language
ADM Architecture Development Method
CHOOSE keep Control, by means of a Holistic Overview, based on Objectives and kept Simple, of your Enterprise
DSL Domain Specific Language
EA Enterprise Architecture
EMF Eclipse Modeling Framework
EMFT Eclipse Modeling Framework Technology
EMP Eclipse Modeling Project
GEF Graphical Editing Framework
GMF Graphical Modeling Framework
GUI Graphical
IDE Integrated Development Environments
IS Information System
IT Information Technology
KPI Key Performance Indicator
MDE Model-Driven Engineering
MDA Model-Driven Architecture
MDD Model-Driven Development
MOF  Meta-Object Facility

MMT  Model-to-Model Transformation

MTT  Model-to-Text Transformation

OCL  Object Constraint Language

OMG  Object Management Group

OSS  Open Source Software

PIM  Platform Independent Model

RACI  Responsible; Accountable; Consult; Inform

RCP  Rich Client Platform

RGB  Red-Green-Blue

SME  Small and Medium-Sized Enterprise

TOGAF  The Open Group Architecture Framework

XMI  XML Metadata Interchange

XML  Extensible Markup Language

XSLT  Extensible Stylesheet Language Transformations
List of Figures

2.1 Amsterdam Information Model [Maes et al., 2000] . . . . . . . . . . . . . 9
2.2 History diagram of EA frameworks [Schekkerman, 2005] . . . . . . . . . . 10
2.3 The Zachman Framework [Zachman, 1987] . . . . . . . . . . . . . . . . . . 11
2.4 A simplification of TOGAF’s ADM [Bernaert, 2011] . . . . . . . . . . . . 12
2.5 ArchiMate structure and TOGAF ADM alignment [Iacob et al., 2012] . . 13
2.6 EA approach with the emphasis lying on four dimensions of the business architecture layer [Bernaert, 2011] . . . . . . . . . . . . . . . . . . . . . . 13
2.7 Four integrated models, together representing the CHOOSE approach [Bernaert, 2011] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15
3.1 CHOOSE metamodel [Bernaert et al., 2013b] . . . . . . . . . . . . . . . . 17
3.2 Illustration of a CHOOSE model [Bernaert et al., 2013b] . . . . . . . . . . 19
4.1 Research steps taken by Bernaert et al. [2013b] . . . . . . . . . . . . . . 21
4.2 ISO 9126 quality characteristics [ISO, 2001] . . . . . . . . . . . . . . . . . 24
5.1 The Eclipse Platform, modified from Vermeiren [2008] . . . . . . . . . . . 28
5.2 The Eclipse Modeling Project [Gronback, 2009] . . . . . . . . . . . . . . . 29
5.3 Four-layer MOF-based metadata architecture, modified from Ameller [2010] 32
5.4 A simplified subset of Ecore [Bacvanski and Graff, 2005] . . . . . . . . . 33
5.5 The EMF structure [Vermeiren, 2008] . . . . . . . . . . . . . . . . . . . . 34
5.6 GMF workflow [GMF] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35
5.7 GMF Dashboard View [Gronback, 2009] . . . . . . . . . . . . . . . . . . . 36
5.8 EMF tree-based editor and EcoreTools diagram editor . . . . . . . . . . . 36
5.9 Four levels of the GMF DSL development [Seehusen and Stolen, 2011] . . 39
6.1 Ecore Model Editor View . . . . . . . . . . . . . . . . . . . . . . . . . . . 44
6.2 Ecore Diagram View . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 45
6.3 CHOOSE Ecore metamodel ........................................... 46
6.4 Node properties view ............................................. 46
6.5 Simplified Graphical Definition Model .......................... 49
6.6 Tooling Definition Model ........................................ 51
6.7 Link constraints for the Supervision relation .................. 53
6.8 Simplified Generator Model ...................................... 54
6.9 Errors during validation caused by violated constraints ....... 55
6.10 Integrated example made with the CHOOSE editor .......... 55

7.1 The project model hierarchy for the brewery case ............... 57
7.2 Blank CHOOSE workspace ....................................... 58
7.3 The editor only allowing viable relations ....................... 59
7.4 Creating new elements using the Connector ................... 60
7.5 Adding Note Attachments ....................................... 60
7.6 Separate CHOOSE viewpoint models ........................... 61
7.7 Linking elements from different models to each other ....... 62
7.8 the tyrecenter example from Bernaert [2011], modelled with the CHOOSE editor ........................................... 63
7.9 Brewery AS-IS model .............................................. 64
7.10 Brewery TO-BE model ........................................... 65
7.11 Brewery AS-IS model with the KIELER layout algorithm .... 66
7.12 Model comparison AS-IS/TO-BE ................................. 67

A.1 CHOOSE Node Concepts. Definitions from Bernaert et al. [2013b] .... 85
A.2 CHOOSE Relation Concepts. Definitions from Bernaert et al. [2013b] ... 86
List of Tables

8.1 CHOOSE editor code size .................................................... 70
Chapter 1

Thesis Overview

The rapid pace of change is a determining factor in every industry. Enterprise Architecture (EA) is currently a hot topic which presents solutions to react to these industry changes. However, although Small and Medium-Sized enterprises (SMEs) would also benefit from adopting EA, a SME-specific approach is still lacking.

This thesis is composed of three parts: the first part provides the reader theoretical insight into the context of this thesis. The second part concerns the tool support development for a new EA approach which is targeted towards SMEs, whereas the third part evaluates the developed tool support.

Throughout the report, the roles Toolsmith and Practitioner are mentioned to distinguish between those who develop tools and those who use these tools in practice.

1.1 Introduction

- Chapter 2 gives an introduction of the general context: bringing EA to SMEs. The chapter is divided into three sections, covering Enterprise Architecture, Small and Medium-Sized Enterprises and last, EA in an SME context.
- Chapter 3 presents CHOOSE, an SME-tailored EA approach. The CHOOSE approach, its metamodel and viewpoints are reviewed.
- Chapter 4 discusses the need for IS tool support to increase chance of adoption and increase the validation of both the tool support as the approach itself. A number of criteria are set which serve as tool requirements upon which the tool support will be evaluated.
1.2 Tool Support Development

- In Chapter 5, an in-depth review is conducted concerning the platform and frameworks used to develop the CHOOSE tool support. Furthermore, the choice of providing Eclipse tool support is well motivated.

- Chapter 6 provides the reader with a technical walkthrough how the CHOOSE editor is developed. The process from starting with a metamodel up to the automatic code generation are fully discussed.

- In Chapter 7, the editor is employed in practice. Instructions on how to employ the CHOOSE editor and conclusions from self-evaluation and case studies are provided.

1.3 Evaluation and Conclusion

- In Chapter 8, the author takes a critical view on the developed tool support and evaluates the tool based on the criteria that were set in Chapter 4.

- Chapter 9 declares the research restrictions this project was subjected to, the future research steps that should be taken in order to optimize the editor and finally a general conclusion.
Part I

Introduction
Chapter 2

Bringing Enterprise Architecture to SMEs

This first chapter provides a descriptive introduction of the context and purpose of this thesis project. As the goal of this thesis is to develop tool support for a new EA approach targeted towards SMEs in the form of a graphical editor, an introduction into of the various concepts are provided.

The chapter is divided into three sections, explaining EA, SMEs, and their relation to one another. As it is not the main goal of this thesis to provide a detailed literature study concerning enterprise architecture, the descriptions are concise. For a more comprehensive view on bringing EA to SMEs, the reader is referred to the work of Bernaert et al. [2013a], who have already performed significant research into this subject. As a result, many of the references made in this chapter point to their research.

The first section provides a declaration of what exactly defines an SME. Bernaert et al. [2013a] define SME characteristics related to Information Technology (IT) adoption, problems that these SMEs encounter as result of a lack of overview of their enterprise, as well as characteristics that should be followed when developing information System (IS) techniques for SMEs.

The second section gives an overview of what EA is and what added value it offers to a company. As it is claimed to be an approach that helps manage change and complexity, it can provide a solution to the problems SMEs are facing.[Ross et al., 2006, Zachman, 2008] Bernaert et al. [2013a] find that EA frameworks share a common structure of dimensions, which can serve as the basis for a new EA approach.

After giving the introduction on SMEs and EA, the third section covers the aspect of bridging the gap between EA and SMEs. This is where the CHOOSE approach is introduced as an effort to bring EA to SMEs by developing an EA approach that is specifically tailored to fit the SME context.[Bernaert et al., 2013a] This approach is then further discussed in Chapter 3.
2.1 Small and Medium-Sized Enterprises

2.1.1 What are Small and Medium-Sized Enterprises?

Small and medium-sized enterprises are a vibrant and growing sector in most economies around the world. [Levy and Powell, 2004] There is no doubt as to the importance of SMEs, as they are an important driver of the economy and the major source of employment. In Europe, they represent 99.8% of the total number of enterprises, and provide around 75 million jobs, or 70% of the total employment. [European Commission, 2012] The European commission defines an SME as an enterprise having less than 250 employees, a turnover of less than € 50M or a balance sheet total of less than € 43M. [European Commission, 2003]

Levy and Powell [2004] indicate that SMEs are not a homogeneous group, but are heterogeneous with diverse needs and objectives. When developing IS techniques, different approaches are needed for enterprises with different characteristics. [Levy and Powell, 2004] As SMEs are the selected target group in this study, it is necessary to understand their characteristics and drivers of growth. Bernaert et al. [2013a] mention characteristics on which SMEs can be distinguished, such as growth, whether the company is family owned or not, influence of the CEO and education of the CEO.

2.1.2 Information Technology Adoption in SMEs

Although IS and IT in large firms provide major opportunities for added value through exploitation of information, there is less evidence of SMEs investing in these matters to capture similar benefits. [Levy et al., 2002] This weaker degree of IT adoption indicates missed opportunities, as the exploitation of information and knowledge is documented as one of the modern times key factors to competitive success for SMEs. [Lybaert, 1998, Hsu et al., 2007]

It is understandable though, as IT adoption is in general more challenging for small firms. Specific challenges are formed by their small size and unsophisticated deployment of information systems. [Bidan et al., 2012] The difficulties faced are caused by a lack of technical competence and know-how, shortages of human resources, risk-aversion, insufficient support technologies, or organizational structures. [Blili and Raymond, 1993, Ballantine et al., 1998, Levy and Powell, 1998, Duhan et al., 2001, Themistocleous and Chen, 2004]
From previously mentioned and other SME characteristics found in literature studies, Bernaert et al. [2013a] list six well-documented SME characteristics that influence their IT adoption:

1. Employees and management are constrained by time, leaving little time to look at strategic matters.

2. SMEs have limited IT knowledge and technical skills.

3. SMEs have fewer resources. The smaller the company, the fewer resources it has, for instance to hire experts.

4. There is a big demand in SMEs for knowledge regarding the performance of tasks and how things are done.

5. The manager or CEO, who is often the company owner, is commonly the single person who decides on strategic issues.

6. In order to accept a new IS approach, the expected returns must exceed the expected risks and costs.

It is clear that these SME-specific characteristics differ from those of larger companies, and thus an adapted IT approach is preferred. Section 2.3 introduces criteria on which IS development, targeted towards SMEs, should be based upon, taking into account the six SME-specific characteristics.

2.1.3 SME Challenges

Enterprises have to be flexible and be able to adapt in response to a changing competitive environment. Globalization, the evolving roles of IT in enterprises and compliance regulations are just a few of the many challenges that force enterprises to adapt in order to survive.[Proper and Greefhorst, 2011]. In response to these challenges, the enterprise, large or small, thus evolves continuously. As a result, its structures change accordingly. Without a certain means of control, structural changes can lead to an overly complex and uncoordinated environment that is hard to manage.[Proper and Greefhorst, 2011] A good example are growth-oriented enterprises. Growth is one of the most common objectives of SMEs, though growth may even present itself without being planned. Among other, growth-oriented companies enter new markets, hire additional personnel, increase the number or scope of their processes and expand their applications and technology infrastructure to support them. The focus on its core objectives, the ability to provide clear personnel job descriptions and asset management can soon be lost. Therefore, companies with a high growth rate need a clear vision providing the company’s strategic direction.[Levy and Powell, 2004]
Although many new firms start up each year, survival becomes increasingly difficult. [Levy and Powell, 2004] Because of the many challenges lying ahead of them, only 70 percent of SMEs survive at least 2 years, 50 percent at least 5 years, a third at least 10 years, and only a quarter stay in business 15 years or more. [Bureau of Labor Statistics US, 2011, Census Bureau US, 2011, Bernaert et al., 2013a]

According to Hambrick and Mason [1984], Kubr [2002], company survival success is conditional on the knowledge of the entrepreneurs. Literature agrees on this importance of knowledge management as one of the sources of sustainable competitive advantages. [Lybaert, 1998] In small and medium-sized enterprises, these entrepreneurs mostly present themselves in the form of one person, combining both the ownership and managerial function. [Omerzel and Antoncic, 2008] It is this person’s responsibility to steer the company in the desired direction by adapting processes to pursue this direction, and to provide clear communications with employees so that the entire company is focused on reaching the desired direction.

Some of the problems that can explain the high SME failure rate can be related to a weakness of the entrepreneur’s knowledge management, causing a lack of structure and overview of the company. [O’Gorman, 2001, Proper and Greefhorst, 2011, Bernaert et al., 2013a] Bernaert et al. [2013a] lists a number of problems that can occur caused by such a lack of overview:

- Although the entrepreneur possesses the most knowledge and knows the overview of the company, communication to employees regarding strategy and structure are difficult without having an explicit overview.

- Concrete job descriptions and overview of tasks and responsibilities of employees are difficult to keep track of.

- As strategy nor processes are static, keeping the processes in line with the company’s strategy is difficult to achieve.

- Impact analysis can help prevent problems occurring. E.g. if an employee leaves the company, what impact does that have on various processes and objectives?

- As SMEs have different stakeholders with their own desires and goals, balancing these goals is a challenging task.

- If the CEO leaves the company, knowledge about the overview of the company has to be transferred to the new CEO.

The following section suggests an approach to help increase a company’s structure and ability to keep a clear overview and as such possibly solving these problems.
2.2 Enterprise Architecture

2.2.1 What is Enterprise Architecture all about?

Enterprises need information systems to function, as IS supports a company’s business processes. Over the years, the complexity of IS has increased a great deal. As a result, it becomes increasingly difficult for enterprises to design their organization, processes, information and IT as a coherent whole without proper support. [Schekkerman, 2004]

To help manage the increasing complexity, Enterprise architecture was introduced as a principle that can be used to describe an enterprise through descriptive representations and can serve as a holistic approach to reduce the system complexity and improve the business alignment of an enterprise. [Zachman, 1997] Note that an enterprise in this context is any collection of organizations that has a common set of goals. In that sense, an enterprise can be the whole organization, a division, business unit or a network. [Schekkerman, 2004]

As IT systems also grow in complexity, the need for Business-IT alignment presents itself. If correctly aligned, IT is applied in an appropriate and timely way in harmony with business strategies. Business-IT alignment, both at the strategic and operational level, was introduced by Henderson and Venkatraman [1993] as the Strategic Alignment Model. This model was expanded to form the Amsterdam Information Management Model [Maes et al., 2000], shown in Figure 2.1.

The expansion adds an Information/Communication column as a bridge between the Business and Technology layer. A structure row is added as the bridge between strategy and operations. It is in these concepts that EA can be positioned, as depicted by the grey area. Some emphasize the use of EA to align IT with the business, others see it broader and use it to keep the processes aligned with the strategy as well. [Bernaert et al., 2013a] Lankhorst defines Enterprise Architecture as:

“A coherent whole of principles, methods, and models that are used in the design and realization of an enterprise’s organizational structure, business processes, information systems, and infrastructure.” [Lankhorst, 2009, p.3]

Through these principles and models, EA captures the essentials of the business, IT and its evolution, which are much more stable than the specific solutions found at hand. The most important EA quality is that it provides the ability to create and maintain a holistic view of the enterprise. [Lankhorst, 2009]

In literature, the discussion often appears whether EA is a product rather than a process. Lankhorst [2009] states it to be both. On the one hand, as a product it serves to guide the organization in a way that keeps it in line with business objectives and policies.
On the other hand, as companies are dynamic, architecture needs to be maintained, and EA should be applied as a process. By treating EA as a process, the organization would obtain greater benefits, as it would increase the organization’s flexibility. This flexibility serves as a competitive advantage, by providing the organization with the capability to adapt faster to changes and stay ahead of their competitors. [Ross et al., 2006, Van Acker, 2012]

### 2.2.2 Existing Frameworks

Numerous EA frameworks have already been developed, differing both in goal and approach, while even more frameworks are to follow. [Roose et al., 2013] A history diagram of frameworks is shown in Figure 2.2, though far from exhaustive. It is clear that most frameworks have a common foundation framework and are built as refinement or add-on of other frameworks. The analysis of different frameworks and their characteristics has already been widely documented, and are not the goal of this section. For an overview of the different framework and comparisons between them, the reader is referred to the studies of Van Acker [2012], Urbaczewski and Mrdalj [2006], Leist and Zellner [2006], Schekkerman [2004]
Several EA techniques are shortly described to prove that there exists a common structure among frameworks, which will eventually serve as the basis for a new EA approach.

**The Zachman Framework**

The Zachman Framework is probably the oldest and best known EA framework. As seen in Figure 2.2, many EA frameworks are derived from this framework. It is depicted by a matrix based on two dimensions. The horizontal dimension declares six stakeholder perspectives, the vertical dimension providing six descriptive focuses that divide the categorization of different EA concepts. The Zachmann Framework can be used only as a classification framework though, as it offers no actual method which describes how to develop an EA. Therefore, it is often used in conjunction with TOGAF, which does specify such a development method. [Bernaert et al., 2013a]

**TOGAF**

The Open Group Architecture Framework (TOGAF) provides a methodology and the required tools to develop and manage an EA, based on the Architecture Development Method (ADM), which specifies an iterative method for developing organization-specific enterprise architectures (see Figure 2.4). The ADM recognizes three types of architectures: A business architecture which supports an agreed upon architecture vision is first to be developed, followed by the information system architecture, which can be divided into a data & application architecture and last, the technology architecture.
Archimate, an Open Group Standard, is an Enterprise Architecture modeling language, providing a uniform representation of EA models, that is widely supported by tool vendors and consulting firms.\cite{Iacob2012} The ArchiMate language consists of three categories: the Passive structure, Behaviour and the Active structure. These categories can be respectively related to the What, How and Who focus concepts of Zachman’s Framework.

Each of the three categories is divided in three layers: a Business, Application and a Technology layer, which are similar to the architecture layers found in TOGAF. ArchiMate was developed to provide a high level approach that can model both the global structure within a certain domain as well as the relations between different domains. This implies that elements from the business architecture layer can be also be linked with elements of the application and technology layers. As of Archimate 2.0, the language is fully aligned with TOGAF (see Figure 2.5), by including a Motivational extension, relating to Zachman’s Why-focus.

ArchiMate is supported by many commercial as well as open source tools. Archi is an example of an open-source editor used to create ArchiMate models. It has been widely adopted and is becoming the de facto open source ArchiMate modeling tool.
This tool is worth mentioning, as it is developed using the Eclipse Modeling Framework, the same framework that will be used to develop the Eclipse-based tool support.

Bernaert et al. [2013a] state that in trying to be as complete as possible, the previous approaches are difficult to implement. Extensive training and certification are needed to correctly apply these approaches. This is confirmed in the EA survey of Buckl et al. [2009], where EA architects conclude that although many organizations first look at TOGAF for Enterprise Architecture Modeling, as it constitutes the most prominent approach, the major drawback is the need to study its very extensive documentation, covering around 800 pages.

2.2.3 A Common Structure

Most EA techniques work with three layers of architecture models, namely a business architecture, applications and data, and technology architecture layer (Figure 2.4). [Bernaert et al., 2013a] As EA was introduced to provide a solution to the growing importance and complexity of IT systems, until now the emphasis mainly concerned the IT aspect, and its alignment with the business. [Henderson and Venkatraman, 1993]

Bernaert et al. [2013a] distinguish four focuses of the Zachman Framework that are shared by almost any EA techniques, namely a strategic (Why), an actor (Who), an operation (How), and an object dimension (What). An extensive analysis of EA frameworks and their concepts relating to these focuses is performed by Bernaert et al. [2013b]. Bernaert et al. [2013a] propose the four dimensions to be used as the basis of a new EA approach targeted towards SMEs. He states that as the business architecture is the foundation for a good EA and of growing importance, it is justified to put the emphasis of the approach on the business architecture layer.
This can be visualized by inverting the EA layer pyramid (Figure 2.6). In the survey of Buckl et al. [2009] performed among enterprise architects, there was a consensus that the business aspect is not sufficiently involved in current EA management projects, as the emphasis still lies mostly on IT.

2.3 Enterprise Architecture in an SME-context

Despite the fact that EA is mostly aimed at large companies, it can as well provide benefits for small and medium-sized enterprises.[Bernaert, 2011] However, there are few EA approaches that are adapted to SME-specific characteristics.
From the six characteristics given in section 2.1.2, Bernaert et al. [2013a] derive the following six development criteria that can serve as guidance for developing IS techniques in an SME-context:

1. The approach should enable SMEs to work in a time efficient manner on strategic issues.
2. A person with limited IT skills should be able to apply the approach.
3. It should be possible to apply the approach with little assistance of external experts.
4. The approach should enable easy descriptions of how things are done in the company.
5. The CEO must be involved in the approach.
6. The expected revenues must exceed the expected costs and risks.

Furthermore, Bernaert et al. [2013a] derive five EA approach criteria from the EA definition given by Lankhorst [2009, p.3]

1. Control: EA is an instrument for controlling complexity.
2. Holistic Overview: Capture the essentials of the enterprise.
3. Objectives: Facilitate the translation from corporate strategy to daily operations.
4. Suitable for its target audience, which conforms to the six SME-specific criteria.
5. Enterprise: Optimization of the company as a whole.

Providing a new approach specifically tailored to SME characteristics is a good start, but useless if the approach is not adopted by SMEs. Based on the Technology Acceptance Model [Davis, 1989] and the Method Evaluation Model [Moody, 2003], Bernaert et al. [2013a] define three actions that should be pursued in order to enhance the approach’ adoption.

- **Increase the Perceived Usefulness.** This usefulness can be related to the advantages of EA techniques. Perceived usefulness is influenced by actual effectiveness. To increase the actual effectiveness, the EA techniques have to be implemented in practice, and through feedback further refined to better meet the target group’s requirements.[Bernaert et al., 2013a] The tool must thus applied and validated in practice.
• **Increase the Perceived Ease of Use.** This implies reducing the effort that has to be spent in implementing the EA technique. SMEs have certain characteristics (cf. supra) such as time and skill limitations that make a special effort to adapt EA techniques to an SME context necessary. [Bernaert et al., 2013a] For the tool support, this implies putting an emphasis on user friendliness.

• **From Actual to Perceived Efficacy** by implementing EA techniques in SMEs. This offers the advantages of increasing perceived ease of use and usefulness through feedback, and increases EA awareness through testimonials and word-of-mouth. [Bernaert et al., 2013a]

Keeping in mind both the target audience (SME) characteristics and the EA approach criteria, Bernaert [2011] introduced the CHOOSE approach, an abbreviation for “keep Control, by means of a Holistic Overview, based on Objectives and kept Simple, of your Enterprise”, referring to the five EA approach criteria. The core thought of this SME-context EA approach is visualized in Figure 2.7. The approach is composed of four integrated Viewpoints, each representing one of the four focuses found in any EA approach. The following chapter will further explain the CHOOSE approach, putting the emphasis on the CHOOSE metamodel.

![Figure 2.7: Four integrated models, together representing the CHOOSE approach [Bernaert, 2011]](image-url)
Chapter 3

CHOOSE Approach

In this chapter, we go into the structure and mechanisms of the CHOOSE approach, with the emphasis lying on its metamodel. A description of each element and viewpoint is given to provide the reader more insight into the workings of CHOOSE.

3.1 Metamodel

In software engineering, new techniques and methodologies are often developed with the aim of improving productivity. Model Driven Engineering (MDE) is a software development methodology that is focused on creating abstractions that represent an alternative to the classical code-based development.[Liu, 2010]

In MDE, models are created according to the elements defined in a metamodel. Such a metamodel defines the rules, elements, constraints... Developing a metamodel is mostly a process, as a model instance needs to be validated after which it can be adjusted to better fit user requirements.

As discussed in the previous chapter, CHOOSE wishes to focus on EA essentials which can be found in any EA technique. This includes four focuses (Why, Who, How, What) and a blend of three layers (Business, Application, Technology). Bernaert et al. [2013b] took the KAOS metamodel as a start from which to refine the CHOOSE metamodel, as KAOS had a metamodel which met the EA essentials.[van Lamsweerde, 2003] Through several rounds of action research, adjusting and refining the metamodel to better meet the SME-context needs, the latest version of the CHOOSE metamodel is shown in Figure 3.1. [Bernaert et al., 2013b] As the action research is still ongoing, the metamodel is still susceptible to changes.
CHOOSE employs four viewpoints, representing the four EA essential focuses:

1. *Why focus* - Goal Viewpoint
2. *Who focus* - Actor Viewpoint
3. *How focus* - Operation Viewpoint
4. *What focus* - Object Viewpoint

The viewpoints are connected to each other through various relations, shown in the metamodel. For the reader to comprehend the basic mechanisms of the CHOOSE, a short description of each viewpoint is given, drawn from Bernaert et al. [2013b]. A list of entity and relation definitions along with the graphical notation elements of the resulting CHOOSE editor can be found in Appendix A.

![Figure 3.1: CHOOSE metamodel [Bernaert et al., 2013b]](image)

### 3.1.1 Goal Viewpoint

*Goal* is the central concept of the goal viewpoint and has *Name* and *Description* attributes. A *Goal* can have a *Conflict* relationship with zero or more other *Goals*. *Goals* can be *Refined* into one or more *Goals* through a *Refinement*. *Goals* can have zero or more *Refinements*. The link between the *Refinement* and the upper *Goal* implies an *OR-Refinement*, while the link between the *Refinement* and lower *Goal* implies an *AND-Refinement*. A *Refinement* must have one, and only one upper *Goal*, and must have one or more lower *Goals*. This implies that each *Refinement* can OR-Refine one upper *Goal* into multiple lower *Goals* via AND-Refinements.
A **Goal** can have a **Wish** or **Assignment** relation with zero or more **Actors**, can be **Operationalized** by zero or more **Operations**, and can have a **Concern** relationship with zero or more **Objects**.

### 3.1.2 Actor Viewpoint

**Actor** is the central concept in the actor viewpoint and has the attributes **Name** and **Description**. An **Actor** can be an aggregation of other actors through a **Division** relation. An **Actor** can optionally be specialized in either a **Human Actor**, a **Role**, a **Software Actor**, or a **Device**. **Human Actors** can be linked through a **Supervision** relation and **Human Actors** can **Perform Roles**.

An **Actor** can have a **Wish** or **Assignment** relationship with zero or more **Goals**, can have a **Performance** relationship (a RACI or other type of performance relation) with zero or more **Operations**, and can **Monitor** or **Control** zero or more **Objects**.

### 3.1.3 Operation Viewpoint

**Operation** is the central concept in the operation viewpoint and has the attributes **Name** and **Description**. An **Operation** can be an aggregation of other **Operations** through an **Include** relation. An **Operation** can optionally be specialized in either a **Process** or a **Project**.

An **Operation** can **Operationalize** zero or more **Goals**, can have a **Performance** relationship with zero or more **Actors**, and can have zero or more **Objects** as **Input** or **Output**.

### 3.1.4 Object Viewpoint

**Object** is the central concept in the object viewpoint and has the attributes **Name** and **Description**. An **Object** can optionally be specified in either an **Entity**, **Actor** or **Association**. This part of the metamodel differs somewhat with the metamodel employed for the CHOOSE Eclipse editor, as further discussed in Chapter 6.

An **Association** Links two **Objects**, while an **Object** can have zero or more **Associations** with one other **Object**. An **Association** inherits the attributes **Name** and **Description**, which are additionally also visualized to be clearer, and a **Link** has the optional attributes **Role** and **Multiplicity**. An **Association** can optionally be Specialized in either (disjoint or) an **Aggregation** or a **Specialization**.

An **Object** can have a **Concern** relationship with zero or more **Goals**, can be **Monitored** or **Controlled** by zero or more **Actors**, and can be **Input** or **Output** for zero or more **Operations**.
Chapter 3. CHOOSE Approach

For a more detailed explanation of the CHOOSE elements, the reader is referred to Bernaert et al. [2013b].

3.2 CHOOSE Integrated View Model

Figure 3.2 shows the example of an SME CHOOSE model that illustrates most of the CHOOSE entities and relations. It is also used in this thesis to illustrate certain aspects. Figure 6.10 shows the same model created with the Eclipse graphical editor as tool support for the CHOOSE approach. The next chapters will handle the development of that tool support.

![Figure 3.2: Illustration of a CHOOSE model [Bernaert et al., 2013b]](image-url)
Part II

Tool Support Development
Chapter 4

Tool Support

To develop EA techniques that have a higher likelihood of adoption in SMEs, Bernaert et al. [2013b] defines several research steps that need to be taken (Figure 4.1). The fifth step implies the development of tool support. This chapter discusses tool usage in practice and the benefits it can offer. On the other hand, in order to develop proper tool support, an understanding of certain aspects such as software quality and requirements gathering is necessary. The end goal is to create a graphical editor tool based on the Eclipse platform, which can be used in case studies to help validate the CHOOSE approach. The research of Bernaert et al. [2013b] involves CHOOSE tool support on different platforms e.g. tablets, smartphones,… This thesis concerns providing tool support based on the Eclipse Platform, which is extensively discussed in the following chapter.

Figure 4.1: Research steps taken by Bernaert et al. [2013b]
4.1 Use of Tool Support

Although many EA practitioners start off using drawing tools to document and share their EA, this approach becomes increasingly difficult as artifacts appear in multiple layers. For example, representations of a software application might appear on a diagram depicting a server, a diagram depicting a business process and a diagram depicting the application’s interfaces. Changes to the application might require updates in all locations of this data, introducing additional complexity and opportunities for inconsistency and inaccuracy. [James, 2008]

Lankhorst [2009] defines a number of advantages that are brought forth by adopting EA tools. First, EA tool support helps in standardizing the semantics and notation of architecture models. (For a list of semantics used in the CHOOSE editor see Appendix A.) Second, tools support the design of correct and consistent models through automated constraint checking. [Lankhorst, 2009] For high-level presentations, the management often prefers using simple notations, e.g. realized with MS PowerPoint [Buckl et al., 2009], while such drawing tools cannot support consistency between models. This is discussed in Chapter 6, where model validation through OCL constraints and the metamodel-driven constraints, ensures the consistency of CHOOSE models. Third, tools can support the practitioner in the application of architecture patterns and reuse of components and solutions. Fourth, tools support the comparison of alternatives, by means of AS-IS and TO-BE models, impact analysis and quantitative analysis of models. [Lankhorst, 2009] Model comparison and impact analysis for the CHOOSE editor are discussed in Chapter 7, where an AS-IS/TO-BE analysis is performed on a Belgian SME. Quantitative analysis, e.g. through the specification of Key Performance Indicators (KPIs) however is not yet implemented in the CHOOSE metamodel, nor is it supported in the CHOOSE editor. This restriction appeared in the performed case studies and is consequently an important part of the future research to be performed.

Bernaert et al. [2013a] argues that tool support, if properly developed, can present substantial benefits for EA techniques. It can facilitate the input (actual efficiency) and enhance the output (actual effectiveness) of applying EA. Tool support can present disadvantages as well though, for example when a learning curve or cost implications reduces the actual efficiency. [Bernaert et al., 2013a]

As the approach is subjected to several SME specific characteristics, Bernaert infers that a user-friendly business architecture method and tool is more likely to be adopted by small businesses. [Bernaert et al., 2013a]
4.2 Software Quality

The ISO 8402-1986 standard defines software quality as:

“The totality of features and characteristics of a product or service that bears its ability to satisfy stated or implied needs.” [ISO, 1986]

Software quality models and their metrics may be used to measure quality in many contexts, for instance during the development of the CHOOSE tool support. [Dromey, 1996, Kitchenham and Pfleeger, 1996]

They consist of an extensible catalogue of quality factors, which describe the quality of a software product or software domains. There is a great variety in these catalogues, many of them in the form of standards. One of the currently most widespread software quality standards is ISO/IEC 9126 Software engineering – Product quality Standard. [Botella et al., 2004]

ISO 9126 measures quality by means of six general software quality characteristics, which are then further decomposed into subcharacteristics. It distinguishes three types of quality: [ISO, 2001]

- Internal quality, measured during the development process.
- External quality, measured throughout the testing process.
- Quality in-use, measured by the user’s view of quality.

Figure 4.2 specifies the six quality characteristics and their decomposition into subcharacteristics. A short description of these six characteristics is given:

- **Functionality** – The essential functions of any product or service. The functions are those that satisfy stated or implied needs. [ISO, 2001] These criteria thus relate to the EA approach criteria of Bernaert et al. [2013a], where the tool should successfully apply the EA criteria.

- **Reliability** – The capability of the system to maintain its service under defined conditions for defined periods of time.[ISO, 2001] This concerns the frequency of failure of the editor, recoverability of data after a software failure and the tolerance of software faults.

- **Usability** – The effort needed for use, refers to how easy it is to use a given function. Usability can only exist to a certain degree.[ISO, 2001] This relates to the SME-context of the tool, in that it has to be user-friendly, maximizing perceived ease of use.
### Figure 4.2: ISO 9126 quality characteristics [ISO, 2001]

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Subcharacteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Functionality</strong></td>
<td>suitability</td>
</tr>
<tr>
<td></td>
<td>accuracy</td>
</tr>
<tr>
<td></td>
<td>interoperability</td>
</tr>
<tr>
<td></td>
<td>security</td>
</tr>
<tr>
<td></td>
<td>functionality compliance</td>
</tr>
<tr>
<td><strong>Reliability</strong></td>
<td>maturity</td>
</tr>
<tr>
<td></td>
<td>fault tolerance</td>
</tr>
<tr>
<td></td>
<td>recoverability</td>
</tr>
<tr>
<td></td>
<td>reliability compliance</td>
</tr>
<tr>
<td><strong>Usability</strong></td>
<td>understandability</td>
</tr>
<tr>
<td></td>
<td>learnability</td>
</tr>
<tr>
<td></td>
<td>operability</td>
</tr>
<tr>
<td></td>
<td>attractiveness</td>
</tr>
<tr>
<td></td>
<td>usability compliance</td>
</tr>
<tr>
<td><strong>Efficiency</strong></td>
<td>time behavior</td>
</tr>
<tr>
<td></td>
<td>resource utilization</td>
</tr>
<tr>
<td></td>
<td>efficiency compliance</td>
</tr>
<tr>
<td><strong>Maintainability</strong></td>
<td>analyzability</td>
</tr>
<tr>
<td></td>
<td>changeability</td>
</tr>
<tr>
<td></td>
<td>stability</td>
</tr>
<tr>
<td></td>
<td>testability</td>
</tr>
<tr>
<td></td>
<td>maintainability compliance</td>
</tr>
<tr>
<td><strong>Portability</strong></td>
<td>adaptability</td>
</tr>
<tr>
<td></td>
<td>installability</td>
</tr>
<tr>
<td></td>
<td>co-existence</td>
</tr>
<tr>
<td></td>
<td>replaceability</td>
</tr>
<tr>
<td></td>
<td>portability compliance</td>
</tr>
</tbody>
</table>

- **Efficiency** – Efficiency concerns the relationship between the level of performance of the software and the amount of resources used.[ISO, 2001] Relates to the first SME criteria: providing a time efficient manner to work on strategic issues or knowledge management.

- **Maintainability** – The ability to identify and repair faults within a software component.[ISO, 2001]

- **Portability** – Refers to how well the software can adopt to changes in its environment or with its requirements.[ISO, 2001]
4.3 Requirements Analysis

To provide successful tool support for the CHOOSE approach, certain criteria have to be met. On the one hand, the tool support would benefit from optimizing the software quality characteristics as mentioned above. As these software quality characteristics are applicable to any software system, and do not entail specific criteria as to the domain context, the targeted audience and its general purpose, more targeted tool requirements need to be determined.

Naturally, the Toolsmith will try to optimize the quality characteristics. However, as Eclipse is the platform used for our tool development, the performance of many of these characteristics are out of our control. As a result of the model-driven approach (see 5.3.1), the focus lies on the code generation through the use of models instead of manual code programming. This causes a high degree of dependency on the Eclipse code generation capabilities to provide us with high software quality. Reliability and maintainability are two major components where using the model-driven approach takes away much control from the Toolsmith. For instance, part of the automatically generated code is overly complex or even redundant, having a negative effect on the maintainability quality aspect, as maintainability correlates with the project complexity.

The more important requirements for the CHOOSE editor are context-specific though. SME-specific characteristics were first mentioned in section 2.1.2, from which six tool requirements can be specified [Bernaert et al., 2013a]:

- Provide a time efficient manner to work on strategic issues.
- A very low IT knowledge barrier, ensuring people with limited IT skills can apply the approach.
- Easy to learn and small effort in getting familiar with the IS without the need of outside help.
- Capture knowledge in the firm regarding performance of tasks and how things are done.
- Enable easy knowledge sharing across the firm.
- Low cost of entry and ensure benefits exceed the costs.

Furthermore, the five EA approach criteria from section 2.3 are also pursued. Bernaert et al. [2013a] have already evaluated the CHOOSE approach in regard to these criteria. If the tool support is successfully developed and correctly aligned with the CHOOSE approach, the author should come to the same conclusions for these criteria.
• Help control the complexity of the enterprise and its processes and systems.

• Provide a holistic overview of the enterprise.

• Facilitate the translation from corporate strategy to daily operations through objectives.

• Suitable to SMEs (cf. supra)

• Interaction of organizational components across different domains.

Chapter 8 deals with evaluating the CHOOSE editor keeping these criteria in mind.

Although a number of criteria have been set, requirements gathering is not a static process that is determined from day one. Through several rounds of action research performed by Bernaert et al. [2013b], a number of requirements have already been defined for the CHOOSE metamodel. However, it may still be further extended to better meet additional user needs after additional validation. In the same way as the metamodel, the CHOOSE editor needs to be adjusted through multiple rounds of validation. Requirements gathering is the iterative process of gathering data, analyzing the data, extract requirements, validating the approach, receiving feedback, adjusting requirements and start validating again.

Even though the CHOOSE editor is developed for an SME-specific context, the approach could also prove to be useful for a larger company. For instance, James [2008] recognizes the need for more basic open-source, low-cost EA tool offerings. He states that they can be used as an introduction to EA management, serve as a repository to hold and manage EA artifacts for presentations and basic models, or can be used as an additional tool that offers basic modeling capabilities, to then be exported and imported into a higher-end, mature EA tool which offers broader analysis capabilities. The latter functionality has already been explored by Roose et al. [2013], where the authors developed a mapping project to convert CHOOSE models to ArchiMate models and vice versa.
Chapter 5

State of the Art

As part of the CHOOSE approach research, tool support is provided for a number of platforms. In this chapter, we motivate the choice to provide Eclipse-based tool support as one of those platforms. As argued in Seehusen and Stolen [2011], creating graphical editors requires customization to a considerable degree. An implication is that the Toolsmith needs to have an understanding of the technologies used to create a graphical editor. We thus go into the Eclipse Platform and the main frameworks that were used to build the CHOOSE-editor, namely the Eclipse Modeling Framework (EMF) and the Graphical Modeling Framework (GMF). Note that the CHOOSE-editor is a Domain-Specific-Language (DSL), as the editor is often referred to as DSL.

5.1 The Eclipse Platform

The Eclipse Platform is an open source, multi-language software development environment, which is the main underlying framework that is used to build our graphical editor. In short, it is a framework for building integrated development environments (IDEs). [Steinberg et al., 2008] As most of the code is Java-based, Eclipse projects run on almost any operating system. [Eclipse]

Eclipse has an extensive and modular plug-in architecture. With the exception of a small run-time core, everything in Eclipse is a plug-in.[Gronback, 2009] This structure makes it possible to extend the functionalities by adding additional plug-ins. For example, one can simply add plug-ins to develop projects in C++, PHP, Perl, and many other languages. During this project, several additional plug-ins were installed, e.g. to provide model comparison capabilities (see Appendix B). The platform comprises a user interface and a workbench. The Eclipse base workbench and its user interface already pack a number of standard functionalities, such as an editor, debug tool, validation capabilities and many more. Nonetheless, for the development of the CHOOSE editor, a number of additional plug-ins were required.
For more details on the Eclipse platform structure and plug-ins, we refer the reader to the very detailed work of Clayberg and Rubel. [Clayberg and Rubel, 2008].

Besides being a platform to develop IDE’s, Eclipse is also a community thriving on its developers to keep extending the framework by developing additional plug-ins. In supporting this goal, Eclipse steers towards an environment that is fully organized to facilitate the free exchange of technology and ideas. [Shavor et al., 2003]

Figure 5.1 situates the Eclipse Platform in regard to other plug-ins. At the top, built upon the GMF framework, which is discussed in the following sections, sits the CHOOSE editor, the Eclipse tool support for the CHOOSE approach.

![Figure 5.1: The Eclipse Platform, modified from Vermeiren [2008]](image-url)
5.2 The Eclipse Modeling Project

The Eclipse Modeling Project (EMP) is a collection of projects related to modeling and Model Driven Development (MDD) technologies, such as Model-to-Model and Model-to-Text Transformation capabilities. MDD is simply put a development approach that uses models as the primary artifact of the development process. EMP was formed to coordinate and focus MDD capabilities within Eclipse. [Gronback, 2009] Figure 5.2 gives a sense of the EMP structure, with at the core the Eclipse Modeling Framework (EMF). EMP provides the projects needed for the development of our DSL syntax. A DSL syntax can be divided into an abstract syntax and a concrete syntax. This separation is required, as a DSL’s superficial structure may be completely different from its underlying structure. [Kleppe, 2007, Krahn et al., 2007]

![Figure 5.2: The Eclipse Modeling Project](Gronback, 2009)

For the abstract syntax, we turn to EMF, which provides us with the necessary development capabilities. The abstract syntax is the central element of any Domain-Specific Language (DLS), which functions as the pivot between concrete syntax and semantics. It represents data that is independent of the physical representation of the data.
We express this abstract syntax by means of a metamodel composed with EMF. All that follows in the further development of the editor, including the development of the concrete syntax, which concerns the representation of the data, then builds on this metamodel.

A concrete syntax can be either textual or graphical. [Fondement and Baar, 2005] In this thesis, we create a graphical concrete syntax, using the Eclipse Graphical Framework.

EMF and GMF are the two main frameworks used in this thesis, and therefore will be described in the following sections. We discuss the different models these frameworks create in this chapter a first time, providing background information how they are created and to understand their function. In chapter 7, a practical approach to the models is given, describing how they need to be leveraged to provide our desired tool features. As EMF and GMF have many purposes and numerous features, we refer the reader to Steinberg et al. [2008] for detailed EMF documentation. On Eclipse fora, this book is often referred to as the bible of EMF. For more detailed documentation on GMF, we refer the reader to the work of Gronback [2009] for an in-depth review on GFM. It is recommended for any Toolsmith creating a GMF-based editor to consult these two books.

5.3 Eclipse Modeling Framework

EMF provides a modeling framework and code generation facility for building tools and other applications based on a structured data model. The two main functions of EMF are mentioned in this description. First it is an environment for modeling and building tools, and second it can be used as a tool for automatically generating code. EMF allows us to define the abstract syntax for the CHOOSE editor through an Ecore metamodel.

There are several advantages to using EMF. First, since it can generate code from an Ecore model to create an Eclipse editor plug-in, Antkiewicz and Czarnecki [2004], Guntli [2010] agree that using EMF significantly reduces the development time to generate code. EMF also makes a domain model explicit, helping to maintain clear visibility of the used model.

As the author of this thesis does not possess a great deal of programming knowledge, the main advantage of using EMF is that it allows the Toolsmith to develop software by modeling instead of programming. Generating code from models belongs to the Model-Driven Architecture (MDA) philosophy. MDA is a concept of growing importance, as many see it as the future of software development.[OMG]
5.3.1 Model Driven Architecture

MDA is a new approach to software design for the development of software systems, proposed by the Object Management Group (OMG) which falls under the category of MDD. It separates and relates platform-independent and platform-specific models using transformation techniques. In short, it thus enables software development based on the transformation of models. IBM emphasizes the importance of models as a way of raising the level of abstraction at which software practitioners understand and build software solutions.

The goal of MDA is to allow the development and management of the whole application lifecycle. This is performed by putting the focus on models so that one can derive code form a stable model as the underlying infrastructure changes over time.[Soley, 2000, Moore et al., 2004] The main difference with other MDD initiatives is that the MDA approach requires technologies that implement the stated OMG standards, such as UML, XMI and BPMN.[Gronback, 2009] The Modeling Project supports many of such standards, making it suitable for MDA.

5.3.2 Models, Metamodels and Meta-metamodels

In a model-driven approach, a clear understanding of the semantics of the modeling language is important, as the models become the most important artifacts in the development of the software.[Seidewitz, 2003] Therefore, it is useful to further describe the models hierarchy employed by EMF.

The manner in which a model is built, is described by a metamodel. How such a metamodel is specified, is in turn declared by the OMG in a meta-metamodel, the Meta-Object Facility (MOF). Models can be divided in four layers. As these different layers are quite confusing, the four layers are illustrated using Figure 5.3.

- **M3** - The layer that contains meta-metadata, describing the properties that meta-metadata can exhibit. As the M3 layer is self-describing, there is no higher layer than M3. MOF is a standard from OMG to describe metamodels. Ecore is a MOF implementation.

- **M2** - The layer containing metadata that describes the properties that metadata may exhibit. Here we find the CHOOSE metamodel.

- **M1** - The layer containing the data of the application, e.g. CHOOSE models.

- **M0** - The reality elements.
5.3.3 Ecore

A snippet of the Ecore metamodel is shown in Figure 5.4, containing the main Ecore elements. The full Ecore metamodel can be found at EMF.

The CHOOSE-editor domain model contains the following four Ecore classes:

- **EClass** – Used to define classes. **EClasses** have a Name, zero or more **EAttributes**, and zero or more **EReferences** to other classes.

- **EAttribute** – Used to represent a modeled attribute. **EAttributes** have a Name and a Type, depicted by the **EDataType** class.

- **EReference** – Used to represent the interrelations between classes. It can have a containment Boolean attribute to indicate if it represents an aggregation or an association. A recursive relation can be created by assigning an **EOpposite** EReference. Note that in Eclipse, composition relations are not available.

- **EDataType** – Used to represent the type of an EAttribute, e.g. EString, EInt,...

As EMF does not depend on the storage format of the model, Ecore models can be created in a number of ways. The output will always be the same though, which is an Ecore model file in XMI format.
Figure 5.4: A simplified subset of Ecore [Bacvanski and Graff, 2005]

Figure 5.5, presenting the EMF structure, shows that there are three ways to create an Ecore model. [Steinberg et al., 2008] The first method is to import from an existing model. This can e.g. be loaded from an XMI file, a UML based model (e.g. a Rational Rose model) or an emfatic model. The second method is the path used in this report, more specifically using the EMF plug-in to define the Ecore model. The third method is to create the Ecore model through Java interfaces.

To define Ecore models, EMF provides a default tree-based editor. A more attractive editor is provided by GMF, in which Ecore Diagrams can be modeled, which are then related to a corresponding Ecore file. The graphical Ecore editor provides useful functionalities such as a tabbed properties view, filter options and many more.

EMF provides a tree-based editor to define an ecore metamodel as described in Bacvanski and Graff [2005], but we will use the Ecore Tools component, a plugin developed by the Eclipse Modeling Framework Technology (EMFT). This plugin renders an environment to create, edit and maintain ecore models. It provides a graphical ecore editor with useful functionalities such as multi-diagram support, a tabbed properties view etc.

5.3.4 EMF Generator model

In addition to the Ecore domain model, which only contains PIM information, a domain generator model (genmodel) is needed to supply the platform specific information. It specifies the code generation options for the domain elements and further decorates the Ecore model.
EMF-generated code is clean, simple and efficient. [Merks and Steinberg, 2005] Code resulting from the GMF framework on the other hand is less efficient and is likely to contain errors.

In most cases, it is not necessary to manually alter the EMF code, though it can be when introducing custom commands. [Seehusen and Stolen, 2011] The Domain Generator Model generates this EMF code in the form of three Eclipse plug-ins, as shown by the bottom layer in Figure 5.5.

- **EMF.model** – containing the Java implementation of the PIM based code.
- **EMF.edit** – containing the UI-independent code, separating the GUI from the business model. It represents code for all possible actions that can be performed on a model element, such as altering a name or simply repositioning a node.
- **EMF.editor** – containing the UI-dependent code, necessary to display the model. The editor code provides the capability to creates instances of the model, making use of the default EMF tree-based editor.

### 5.4 Graphical Modeling Framework

The Graphical Modeling Framework sits on top of the Eclipse Modeling Framework and the Graphical Editing Framework (GEF). GMF is a relatively new project, released in 2006, replacing GEF as the previous framework used to build Eclipse graphical editors.
As GEF and EMF are two independent frameworks, GMF was introduced as an easier way to develop editors by providing an integrated framework that has the GEF capabilities while being linked to an underlying EMF model. [Gronback, 2009, Moore et al., 2004]

In the context of GMF, EMF is used to define the metamodel or abstract syntax, represented by an Ecore model, while GEF is used to implement the concrete graphical syntax. [GEF] For more information on GEF, the reader is referred to Moore et al. [2004], Vermeiren [2008], Gronback [2009].

GMF consists of two main components; a runtime and a tooling framework. [Gronback, 2009] The runtime component handles the task of bridging EMF and GEF, and includes reusable diagramming components providing a number of functionalities to ease the development of graphical editors. Connection handles, toolbars, properties views and diagram export to image files are just a few examples.

![Figure 5.6: GMF workflow [GMF]](image)

The tooling component presents a model-driven approach for generating diagrams. Figure 5.6 shows a clear overview of the different models used in the GMF tooling workflow. GMF comes with a powerful tool to help the Toolsmith create graphical editors, named the Dashboard view (see Figure 5.7). Using the Dashboard, each type of model needed for constructing the graphical editor can be selected, created or edited. The various models are discussed in the following sections.
5.4.1 Domain Model

The first step when creating a GMF project creating or referencing a domain model. The domain model is an Ecore model as discussed in section 5.3.3. As previously mentioned, the default EMF editor for creating Ecore models presents a tree-based view. By installing the Ecore Tools plug-in from the Ecore Modeling Framework Technology (EMFT) project, a graphical editor is supplied which supports .ecorediag files. The two type of views are shown in Figure 5.8. Note that these two views represent the same domain model. From the Ecore model, an EMF Generator Model (.genmodel) needs to be derived. This model then generates the code to create the three Eclipse plug-ins mentioned in section 5.3.3.
5.4.2 Tooling Definition Model

The Graphical Definition Model (.gmfgraph) defines the graphical elements found on a diagramming surface such as shapes, labels and connections. The Dashboard provides a wizard to derive the Graphical Definition Model by letting the Toolsmith determine what domain element to represent as a node, connection or an attribute. As creating an elaborate Graphical Definition requires some effort, it is useful to know that they can be reused in other editors. For example, a set of UML2 figures is already defined by the UML2 Tools Component, and is available to be used in any editor.

5.4.3 Graphical Definition Model

The purpose of the Tooling Definition Model (.gmftool) is to define all the elements that compose the diagram palette, which is used to create and work with diagram content. The utter-right part of Figure 5.8 is an example of such a palette. Same as with the Graphical Model, the Tooling Definition can be derived through the Dashboard, and can be reused in other diagram editors.

5.4.4 Mapping Model

The Mapping Model (.gmfmap), as Gronback [2009] describes, is the heart, and consequently the most important of the GMF models. It combines the previous three models, linking elements from the Domain Model to a graphical representation defined in the Graphical Model, and assigns them to a palette element, defined in the Tooling Model. The Mapping Model thus represents the actual diagram definition. Using the Dashboard, the previously defined models can be combined into the Mapping Model. Eclipse seldom correctly maps the elements to each other though, so the model has always be reviewed to ensure that the domain, graphical and tooling elements are correctly mapped.

As it is the place to declare many other features, manual modifications are frequent in this model. Adding labels to nodes and links is one possibility. A number of different label types can be used. Feature Label Mappings are used to represent an EAttribute that is to be defined by the practitioner, such as element names. As they are part of the metamodel, the labels are stored in the domain model. As opposed to Feature Labels, normal Labels can be used to display read-only information. Design Labels are employed to display information that is not linked to an attribute of the metamodel and is consequently not stored in the diagram model.

Link constraints are also defined in the Mapping model. For each Link, Source and End Constraints can be added. GMF recognizes a number of constraint languages, including Object Constraint Language (OCL), Java and Literal.
In the Mapping Model, palette elements can be assigned to multiple model elements. It is thus allowed to provide one palette element for multiple link connections. When connecting different model elements, GMF determines the possible relations between the model elements as determined in the Ecore model. If there are multiple relations between the elements, GMF creates a pop-up menu presenting the choice between all valid options.

5.4.5 Generator Model

The GMF Generator Model (.gmfgen) is the largest model used in GMF, and the one most likely to be extended to provide customizations.[Gronback, 2009] It is composed of several elements which are highly customizable. By adding a shortcut feature, diagram models can import elements from other diagrams. This feature will prove to be very useful to the Choose-editor. Additional properties like copyright text, diagram and domain file extensions, a model ID and a dynamic template directory are specified in this model.

The last step in developing the graphical editor involves the Model-to-Text Transformation (MTT) of the Generator Model, which creates the GMF.diagram plug-in. This is simply accomplished by generating the diagram editor using the Dashboard View. Eclipse supports Model-to-Model Transformation (MMT) capabilities which transform the Mapping Model to create the Diagram Generator Model. Eclipse can create Rich Client plug-ins, by marking the Dashboard’s Rich Client Platform (RCP) checkbox, GMF then creates a diagram editor plug-in that runs on a bootstrapped Eclipse configuration, meaning that it strictly only employs the plug-ins necessary in order to run the diagram editor. Note that the .edit and .editor code generated by the Domain Model are still necessary to launch the diagram editor, as they represent the domain information.

Upon the creation of this last plug-in, the diagram editor can be launched. Eclipse by default launches a configuration that includes any plug-in present its workspace. When using Eclipse extensively though, the number of installed plug-ins increases. This can cause the launch configuration to become quite memory-intensive, and it can create a number of errors as a result of plug-ins that are not compatible with each other. By selecting Run→Run Configuration in the menu, a new launch configuration can be created with only the necessary plug-ins selected.
This entire process of EMF and GMF code generation can be summarized using Figure 5.9.

- **Level 0** – The .ecore metamodel which is the basis of the code generation.
- **Level 1** – The configuration models used to create the GMF Generator.
- **Level 2** – The code generator models for both EFM and GMF transformations.
- **Level 3** – The source code, comprised of EMF code, GMF code, modified GMF code and custom code.

5.5 Why the Eclipse Platform and GMF?

In this section, we motivate the choice of the Eclipse Platform and the Graphical Modeling Framework being a good platform to provide tool support for the CHOOSE approach.

5.5.1 Open Source

While there is no universally accepted definition of Open Source Software (OSS), Gacek and Arief [2004] outline the three main criteria points for determining if a particular software is open source:

- The ability to distribute the software freely.
- The availability of the source code.
- The right to create derived works through modification of the code.
OSS has several advantages over commercial software besides having a significantly lower cost. All who desire are offered free access to the source code of OSS, as opposed to commercial software, where only insiders can access that code. [Von Hippel and Von Krogh, 2003] Stol and Ali Babar [2010] report benefits such as the availability of high quality products, adherence to open standards and the absence of vendor dependency.

Raymond [1999], Johnson [2006] state that open source bugs are resolved more quickly than closed source bugs, as more people review the code. The analysis of Kuan [2002] confirms this statement. Consequently, the security aspect of OSS is also augmented, as there is a greater chance to identify security flaws and other problems as the amount of people that can review the code increases. [Lerner and Tirole, 2005] Due to the high amount of people having access to the source code, a great number of 3rd party applications is available. Therefore, the platform has a greater chance to meet the desires of the user.

On the flipside, OSS can entail a number of disadvantages. Stol and Ali Babar [2010] provide a comprehensive list of challenges, from which several were experienced during the course of this thesis project.

### 5.5.2 Development Time and Editor Quality

As previously mentioned in section 5.3, Antkiewicz and Czarnecki [2004], Guntli [2010] agree that the EMF code generation significantly reduces the development time to create a model editor.

Buchmann et al. [2007], Krogmann and Becker [2007], Seehusen and Stolen [2011] state that GMF can significantly reduce development time for developing a graphical editor. Krogmann and Becker [2007] executed a project in both a traditional programming and an Eclipse model-driven manner, and found that the latter could be carried out almost ten times faster than the conventional approach, while simultaneously improving the code quality. This reduction in development time may not be immediately noticed though.

Buchmann et al. [2007], Seehusen and Stolen [2011] agree that the reduction in development time only becomes apparent when the Toolsmith is familiar with the GMF framework. A lot of time is often lost in unraveling the Eclipse code mechanisms. As the GMF generated code often comes short and modifications have to be made, finding the right places in the generated code to implement modifications is time consuming.
Evans2009, Kolovos2010 confirm this by concluding that a Toolsmith requires a great deal of programming experience, Eclipse and EMF knowledge in order to properly use GMF. One of the main causes for this is poor documentation. Holmes and Walker [2008], Seehusen and Stolen [2011] In turn, this is caused by the dynamic characteristic of GMF. Updates and new versions of Eclipse and its frameworks are introduced at a fast rate while plug-ins are often not compatible with previous version.

Amyot et al. [2006] performed a study comparing various DSL tools, and although the Eclipse frameworks were concluded to be the most difficult to use, it resulted in the best editor usability and graphical completeness.

Because of its complexity, several Eclipse extensions exist that simplify the process of DSL development. Kolovos et al. [2010] state that implementing a graphical editor using GMF is a loosely guided and error-prone process, as it requires developers to hand craft complex interconnected models. The authors developed an open source tool, EuGENia, as a plug-in that sits on GMF which helps simplify the tooling process. This tool was explored by the author but the possibilities with EuGENia proved to be far lower, unsuitable to develop the CHOOSE editor.

5.5.3 Academic Praise and Support

Eclipse has considerable support from leading companies and organizations in the technology sector such as IBM and Micro Focus. Among other through academic initiatives such as IBM, Eclipse is actively promoted in an effort to set it the de facto development environment. Eclipse promotes itself as offering value to researchers and educational faculties by providing an industrial-strength infrastructure for conducting research and developing curricula in many areas of computer science and computer engineering, with particular relevance to programming languages, development tools, collaboration and programming environments.

The platform is gaining widespread acceptance in both the commercial and academic world, and is for many developers the preferred environment due to its flexibility. [Steinberg et al., 2008] Performing a search in the ACM digital library for ‘Eclipse’ and ‘Netbeans’, one of the main competitors of Eclipse, yields respectively 6,000 and 500 results. This certainly gives an indication as to its academic popularity. Numerous Eclipse-related academic projects have been developed, including the EuGENia project (cf. supra) and Objectiver, a KAOS modeling tool.
5.5.4 Transformation Capabilities

Bernaert et al. [2013b], James [2008] discuss a possible need of switching between EA techniques. For instance, if a more elaborated EA approach is needed as a company grows and requires more detailed architectures, a switch from the CHOOSE approach to ArchiMate could be made.

Towards this goal, Eclipse supports Model-to-Model Transformation languages, such as the ATLAS Transformation Language project.[Jouault and Kurtev, 2006] In the field of Model-Driven Engineering (MDE), ATLAS provides ways to transform a source model (confirming to a source metamodel e.g. the CHOOSE metamodel) into a target model (e.g. ArchiMate metamodel).

A second transformation capability is XSLT, which is capable of converting XML files to other formats. As all models used in the development of the CHOOSE editor are persisted in XMI, converting CHOOSE models to other languages is certainly possible. As mentioned in section 4.3, Roose et al. [2013] already developed an Eclipse XSLT transformation project which makes mapping CHOOSE to ArchiMate and vice versa possible.
Chapter 6

Technical Walkthrough

This chapter provides the reader with a technical walkthrough which describes the different steps taken to reach the goal of the tool development, the Eclipse CHOOSE editor plug-in, which we refer to as ‘the editor’. These steps correspond to the different aspects first mentioned in section 5.4 and are exemplified using model snippets. The source models can be downloaded as described in Appendix B to look into the full models.

6.1 Domain Model

The process of building a GMF-based graphical editor starts with the Domain Model, serving as an M2 PIM metamodel. The final Ecore metamodel has much evolved from the original metamodel, as it evolved based on new changes to the metamodel of Bernaert et al. [2013b]. Also exploring the capabilities and encountering the restrictions of the Eclipse Platform caused changes to be made to the metamodel. The final Ecore metamodel is shown in Figure 6.3.

To start the tool development, a new project has to be created:

File→New→Other→Graphical Editor Project

This creates a new project with a JRE library, basic.ecore dependencies, a model folder and by default includes the Dashboard View.

An Ecore Diagram is added to this folder, in which we can model the metamodel.

File→New→Other→Ecore Diagram

By selecting the Ecore Diagram from the EMFT component, Eclipse creates two files. First is a .ecore file, the domain model (Figure 6.1). The second is a .ecorediag file, containing the graphical information. The .ecorediag file is based on a GMF-based plugin, providing a Canvas and Palette to create UML-style models, an Outline View and Tabbed Properties (Figure 6.2).
The full Ecore metamodel is shown in Figure 6.3. Notice that this ecore metamodel very much resembles the UML-based CHOOSE metamodel as made by Bernaert et al. [2013b]. There are differences however, of which the implications are described. These differences present themselves either because there are Eclipse restrictions as to the possibilities for generating code, or because of the author’s restricted programming skills.

At the top of the metamodel sits the Diagram EClass, which acts as a the diagram root element. Eclipse requires such a root object, as every EClass that will eventually serve as an artifact in the resulting editor will be stored there. Eclipse must be able to trace back each EClass to this root object through one or more EReferences with a containment property, or through inheritance relations.

The containments are represented by the five aggregation relations that depart from the Diagram EClass to each CHOOSE entity EClass. If this is not the case, Eclipse gives an error describing the existence of ‘phantom’ nodes, i.e. nodes that have no place to be stored.

6.1.1 Goal Viewpoint

The Goal Viewpoint in the CHOOSE Ecore metamodel is nearly identical to that of Bernaert et al. [2013b]. There are small differences, such as the lack of a Refinement ID attribute. This feature posed no functionality in the CHOOSE editor and is purposefully left out.
Chapter 6. Technical Walkthrough

Furthermore, the EReference names were modified. This is because the EReference names are implemented in the properties view in the runtime (see Appendix C). The names need to be specified to how the Toolsmith wishes to see them displayed in the editor.

Unfortunately, GMF does not provide a step in the development process to customize the properties view, thus it must be foreseen in the metamodel. Useful to know is that a capital letter in an EReference name is visualized as a space in the editor. For example, in the domain model, Goal has a ConcernsObject relation with the Object node, which translates into the property name shown in Figure 6.4.

Last, the Name attribute of the Goal entity has given a lower bound of 1. This is applied to each Name attribute of the four CHOOSE Viewpoint entities. It improves the model consistency, as an error message is given for each Node which has no Name specified when the Practitioner chooses to validate the model. (Edit→Validate)
6.1.2 Actor Viewpoint

Bernaert et al. [2013b] model a Division aggregation and Includes aggregation relation in respectively the Actor and Operation viewpoints to depict a containment. In the Ecore metamodel these were not modeled as containment relations because of Eclipse restrictions. The platform specifies the contained element can be part of only one container, which was undesirable in practice. For example, a company has two Actors, Team A and Team B, who are responsible for various Operations. If a Human Actor would be part of both those teams, as he has responsibilities in both teams, a containment relation would not allow this.

Therefore, the author decided it would be better to model normal relations, with the visual look of an aggregation relation, depicted by a diamond shape at the containing node.
The metamodel of Bernaert (2013) describes an Actor entity with an optional specialization of Human Actor, Role, Device or Software Actor. Difficulties were experienced in regard to the modeling of both the Actor and the four specialized Actors. As a result, the metamodel now has an additional EClass to which the Actor class is added as a specialization, implying that the Actor elements is modeled at the same level as Human Actor, Role, Device and Software Actor. The additional class is named Actor-Container as Eclipse does not allow two identical Name strings in a model. Concerning the CHOOSE approach, this solution has no immediate implications, as Eclipse does not distinguish between (non)-specialized nodes. The specialization relation thus simply involves inheriting the Name and Description attributes.

Furthermore, the Performance Type relation between Actors and Operations has been specified by modeling RACI performance relations (Smith and Erwin 2007). A brief definition of the four RACI relations is given:

- **Responsible (R)** – The Actor who must perform the work is responsible for the activity until the work is finished and approved by the Accountable Actor. There is typically only one person responsible for an activity. [Cabanillas et al., 2012] In the case studies, it was often preferred to work at an abstract level, assigning high level Operations to Organizational Roles.

- **Accountable (A)** – The Actor who must approve the work performed by the Actor responsible for an Operation. Literature agrees that there must be one and only one accountable for each Operation. [Smith et al., 2007, Morgan, 2008, Cabanillas et al., 2012] As a result, the Accountable relation has been given a one-to-one cardinality as the side of the Actor. Might an Operation not have an Accountable Actor, the CHOOSE model will not successfully validate the model.

- **Consulted (C)** – This role involves the Actors whose opinion is sought while performing the work, and with whom there is two-way communication. [Cabanillas et al., 2012]

- **Informed (I)** – Actor who is kept up-to-date about the progress of an Operation and/or the results of the work, and with whom there is just one-way communication. There may be more than one informed Actor for an Operation.[Cabanillas et al., 2012] Other types of performance relations exist, e.g. RASCI [Cabanillas et al., 2011], which adds a Sign-off relation, but additional types were not added to the editor for two reasons. First, the current case studies indicated that the RACI relations fulfilled all requirements. Second, to make the final editor as simple as possible, one of the key requirements as to keep a holistic overview is to limit the practitioner to as few artifacts as needed. Omitting one or more of the RACI relations was considered, but were kept to provide the ability to model performance relations to a certain detail.
A last difference is the absence of an Actor optionally being a Specialization of an Object. This was not included because the author could not operationalize an editor with such functionalities.

### 6.1.3 Operation Viewpoint

Differences in the Operation Viewpoint are similar to those of the Actor Viewpoint. There is an OperationContainer class which specializes Operation, Process and Project. The Includes relation is again modeled as a normal relation.

### 6.1.4 Object Viewpoint

The differences that occur regarding the Object Viewpoint are as a result of difficulties attaining the desired modeling effect in Eclipse. It is possible to model a relation as an `EClass`, as is the case for the Association relation in the metamodel of Bernaert et al. [2013b]. However, relations modeled as `EClass` are not included in the properties view. Thus, when selecting an Object in the CHOOSE editor, the practitioner would only be able to graphically see which Objects it is Associated to, as it will not be listed in its properties view. This forms overview problems when a model increases in size, as it becomes increasingly difficult to graphically spot the Association relation. Consequently, the Aggregation, Specialization and Association relations are modeled as normal EReferences. Aggregation and Specialization relations have appropriate visual design, i.e. respectively a diamond shape and a hollow closed arrow.

This does restrict the practitioner as Name or Description attributes cannot be assigned to the relations. This in turn implies less flexibility, as using the Association relation with a specified Name or Description could be used to many ends. In Chapter 7, the possibility of adding notes containing descriptions to a connection is discussed to provide a possible solution to this problem.

Using the Dashboard View, a Domain Generator Model can be derived from the metamodel. The package name, prefix and URI have to be set in this model. For our editor needs, the Multi-line property of each Description `EAttribute` is set to true. This allows an entity’s Description attribute to be entered in a multi-line text field. From the Domain Generator Model, the EMF.model, EMF.edit and EMF.editor code are generated by right clicking on the root element in the .genmodel file and creating the required plug-ins.

### 6.2 Graphical Definition Model

As the automatically derived Graphical Definition Model provides a very simplistic layout, the Graphical Model is totally redesigned.
On the left of Figure 6.5, a simplified version of the Graphical Definition Model is displayed, containing only the visual elements to represent the Goal, Node and a Conflict relation to another Goal. On the right, the elements as they would look like in the resulting editor are shown. GMF does not yet support instances to be created in the Graphical Model, so graphical features can only be reviewed when all steps of the Dashboard are completed. The required custom code modifications are discussed in Appendix C.

The **Canvas chooseeditor element** serves as the container for all diagram elements, consisting of Nodes, Connections, Compartments (of which there are none in the Choose editor) and Diagram Labels. Each element requires a **Figure Descriptor** which defines its graphical characteristics. The **Figure Descriptors** are all contained in the **Figure Gallery**. As mentioned in section 5.4.3, **Figure Galleries** are reusable.

![Figure 6.5: Simplified Graphical Definition Model](image)
The Goal Node is related to the GoalFigure Figure Descriptor which defines the shape of the node as a Rounded Rectangle. The Size children determine the Node’s default and maximum Size. A Red-Green-Blue (RGB) background color is defined, as well as a Grid Layout. A Diagram Label to display the Name EAttribute is added, containing Grid Layout Data, which ensures the label is centered by grabbing the excessive space of its Node. The Name initialization is obtained by adding a Child Access to the Figure Descriptor.

It is worth mentioning that as opposed to the early prototypes of the CHOOSE editor, the connections do not display a label. Self-evaluation and feedback from the case studies showed that adding labels to connections to display the relation type has a negative effect on the ability to keep a clear overview, due to excessive information clutter. Therefore, another approach was pursued, providing separate graphical definitions for each relation.

In previous prototypes, the CHOOSE editor had a separate graphical shape for each entity, based on the shapes in Bernaert [2011]. Custom shapes can be defined by adding a Scalable Polygon and adding Template Points. GMF does not fully support the use scalable polygon shapes though, as it sets the enveloping square as node bounds. It was opted to display each entity as a rounded rectangle as this provided the best design properties.

GMF by default generates connections as a simple black line with an open arrow. To create shape decorations for our connections, a Polygon Descriptor is assigned to the connection’s Figure Descriptor, e.g. the red thunder decoration in Figure 6.5. The shape is defined by adding Template Points, and assigning the Polygon Decoration as a Source or Target Decoration. GMF does not support decorations to be placed in the middle of the connection, but a manual code alteration is provided in Appendix C to force this feature. More information on the different types of properties and elements that can be used in the Graphical Definition Model are described in Gronback [2009], although the properties described are not exhaustive, and few graphical examples are provided.

6.3 Tooling Definition Model

The Tooling Definition Model contains the information concerning the creation tools of the diagram elements. Similar to the Graphical Model’s Canvas, the Tooling Definition Model has a Palette root element which acts as container for all Creation Tools. The left side of Figure 6.6 shows the CHOOSE Tooling Definition Model, with the right side displaying the Palette view in the resulting editor.
When automatically derived via the Dashboard, GMF adds creation tools for each type of Connection. As GMF can detect viable relations between certain nodes and present a choice of relations, there is only one Connection Creation Tool required.

Furthermore, to create a polished look, customized icons are added for each element. To achieve this, a Figure Bundle is added to each creation tool with a classpath property referring to an icon image. The default place for diagram icons is found in the EMF.edit/icons folder. The icons for the CHOOSE editor are custom made but found their inspiration in corresponding Archi element icons. [Archi]

### 6.4 Mapping Model

As described in Section 5.4.4, the Mapping Model links the elements from the Domain, Graphical and Tooling Definition Model. Unfortunately, when automatically deriving the model through the Dashboard, GMF does a poor job in linking elements to each other, so the Toolsmith is recommended to review each element and ensure the correct mapping.
Chapter 6. Technical Walkthrough

A snippet of the mapping model, containing the elements to model an Actor Node and a Supervision relation between Actors is shown in Figure 6.7. As previously mentioned, each Node needs to be traced back to the Diagram root element through a containment relation. This happens in the Top Node Reference, by specifying a Containment Feature. The Node Mapping is the actual bridge between the different models, by linking a Domain Element, Diagram Node and Creation Tool. A Feature Label child is then added to the Node Mapping representing the Actor’s Name Label. It is not necessary to provide a Label Mapping for the Actor’s Description attribute, as it is not shown on the Diagram Canvas, only in the Node’s Properties View.

The Link Mappings represent Connections, for which a Diagram Link from the Graphical Model and a Creation Tool element need to be specified. For the CHOOSE editor, every Link Mapping has the same Creation Tool. This optimizes the tool’s user-friendliness, as the Practitioner automatically gets a list of viable connections when creating a relation between two Nodes. This is illustrated in the following chapter.

As the links used in the Choose editor are all bi-directional, two Link Mappings exist for each type of relation. Providing two Link Mappings improves the flexibility of the tool, by enabling the practitioner to e.g. create a supervision relation Supervisor→Supervisee, but Supervisee←Supervisor as well, upon which the editor will ask in which direction the Supervision relation should point.

Link Constraints can be added to the Connections. Cardinality constraints are automatically persisted through the metamodel. Thus, if an Operation would have an Accountable Actor, the editor does not allow adding another Accountable relation. Eclipse supports various languages for describing constraints. For the editor, several OCL constraints were applied, though it does not cover the more complex OCL constraints from Bernaert et al. [2013b] due to the author’s restricted skills in the matter. The OCL constraints for the Supervision relation are shown in Figure 6.7. The Source Constraint specifies that a closed Supervision loop may not exist. The Target Constraint states that a Human Actor cannot supervise himself. The constraint also includes the inability to create supervision loops. E.g. if Person A supervises Person B who supervises person C, Person C can not be the supervisor of person A. Upon finishing the Mapping Model, the Diagram Generator Model is to be derived.
6.5 Diagram Generator Model

This final model is used to generate the diagramming code. Figure 6.8 shows a snippet of the Generator Model containing the elements for the Actor Node and Division relations. The most important changes are described:

- In the **Gen Editor Generator**, domain and diagram file extensions have to be defined.

- In the **Gen Diagram**, enable the use of Validation by setting the Validation Enabled and Validation Decorators as true, and the validation Decorator Provider Priority to Medium. This allows us to validate the model in the runtime, shown in Figure 6.9, where errors occur caused by a Node with an empty Name and an Operation without an Accountable Actor.

- Shortcut capabilities are added by creating a **Shortcut Action** in the **Gen Editor Context Menu**. These shortcuts are used to implement elements from other diagrams.

- As GMF assigns a default Undefined Name value to each connection, a Display Name has to be specified in each Connection’s **Gen Link Specialization Type**, displayed by the selected element in Figure 6.8.
6.6 Custom Code Modifications

As the generated GMF code can only provide certain functionalities to a certain extent, custom code modifications are mostly required when developing an elaborate editor. In Appendix C, these additional functionalities are further described by means of the source code alterations. Additional functionalities that were included:

- A property sheet in which the attributes and relations are not ordered alphabetically as default by GMF, but in a desired order.
- Multiline text labels
- Gradient color Nodes
- Decorations placed in the middle of a connection
- Circle decorations

Figure 6.10 shows the CHOOSE example model from Bernaert et al. [2013b], made with the CHOOSE editor.
Figure 6.9: Errors during validation caused by violated constraints

Figure 6.10: Integrated example made with the CHOOSE editor
Chapter 7

The Choose Editor in Practice

This chapter describes how to use the CHOOSE editor in practice, to build CHOOSE models and exploit the various functionalities. The steps and functionalities are augmented with comments and conclusions drawn from the performed case studies. The editor’s restrictions and possible future extensions are also discussed.

The editor was validated in two Belgian SMEs, with the goal to obtain feedback and gain more insight into the features that SMEs expect when looking to enterprise architecture. Furthermore, by validating the editor at these SMEs, the CHOOSE approach was implicitly also further validated. Company names are not mentioned due to anonymity reasons.

The first SME was a Belgian governmental enterprise with a complex structure as they have a great amount stakeholders, implying that finding a good balance which goals to strive for is a problematic issue. Furthermore, when hiring new personnel, it takes considerable time for the new employees to fully understand the structure of the enterprise. They saw enterprise architecture as a means to make company knowledge explicit. By modeling their organization structure, company goals, etc. the communication across the company should happen more efficient.

The second SME was a Belgian brewery operating in an international context. They have known a very high growth rate, purchasing new infrastructure, IT systems, etc. whenever it deemed necessary. They looked to enterprise architecture techniques as a means of making easier strategic decisions. These strategic decisions concerned prioritizing their different goals and having a clear overview of what impact certain changes imply.
Chapter 7. The Choose Editor in Practice

7.1 Setup for CHOOSE modeling

To use the CHOOSE editor, a version of Eclipse and additional required features need to be installed. Appendix B covers this installation process. Because the editor is just an Eclipse plug-in, it depends on the Eclipse platform to provide its functionalities. To increase the user-friendliness by eliminating the installation procedure, deploying the editor as a stand-alone executable file should be part of future research.

When installed, a new project should be created File→New→General Project in which Folders can be added to contain CHOOSE models (see Figure 7.1). To create a CHOOSE model, select File→New→Example→ChooseEditor Diagram. An editor project is composed of two files i.e. a domain file serving as repository for the various CHOOSE elements, and a diagram file which contains all the graphical information.

![Figure 7.1: The project model hierarchy for the brewery case](image-url)
Chapter 7. The Choose Editor in Practice

7.2 Modeling Environment

The new, blank CHOOSE workspace is shown in Figure 7.2.

![Blank CHOOSE workspace]

Figure 7.2: Blank CHOOSE workspace

The workspace is divided into the following sub-windows:

- The **Project Explorer** tree window. By default positioned at the top left. This is where the CHOOSE models can be viewed as a tree structure.

- The **Properties** window. This displays the properties of a selected CHOOSE element, including the entities linked to it. Relations and attributes can also be edited in this view.

- The **Outline** window. This window displays the contents of the canvas in miniature as a navigation tool for the selected diagram.

- The **Problems** window. When validating a model, the violated constraints are shown in this window as errors.

- The **Layout** window. This is the resulting window from installing the KIELER automatic layout plug-in (see Appendix B). The properties for the automatic layout algorithm are entered here.

- The **Palette**, containing the drawing tools, and elements and relations that can be added to the canvas.
7.3 Constructing CHOOSE Models

To create new elements and relationships in the CHOOSE canvas, select the required element tool on the palette and click it on the canvas. Note that by holding down the Ctrl key (Command key on Mac), the palette tool remains selected. This speeds up the modeling process.

To add a new relation, select the Connector Palette Tool. The CHOOSE editor only allows relationships that are specified in the CHOOSE metamodel. By providing only one Connector tool, the Practitioner does not need to be familiar with the rules governing CHOOSE relations.

If a certain type of relationship already exists between the source and target elements, the editor will not allow that relation to be mapped again and presents the remaining possibilities. Figure 7.3 illustrates this: the Operation has an Accountable Actor, and as the CHOOSE metamodel only allows one Accountable Actor, that relation is no longer available to another Actor.

![Figure 7.3: The editor only allowing viable relations](image)

Besides using the Connector to create a relation between source and target elements, it can be used to create new elements as well. This was well received and applied mostly when refining Goals. Figure 7.4 illustrates this. Instead of creating three Goals, one Refinement and linking them together, the Practitioner can speed up this process by creating new elements using the Connector. Drawing the Connector from the Refinement
node to the canvas, a new lower level Goal can be created. Note that the option to make an OR-Refinement no longer exists, as the Refinement already has a Supergoal.

Figure 7.4: Creating new elements using the Connector

One of the CHOOSE editor’s restriction in regard to the CHOOSE metamodel is the absence of attributes for associations and the specialization of an Actor as an Object. Consider the situation as illustrated in Figure 7.5, where the senior partner supervises a team. As a Supervision relation can only occur between Human Actors, this could be resolved by creating an Association with a name label. However, as this lacks in the editor, a possibility is to create Note Attachments (available in the Palette) to elements. Node Attachments are only visual though, so the information is not persisted in the domain model. Part of the future research is further developing the CHOOSE Ecore metamodel and Editor so that is fully aligned with the CHOOSE metamodel of Bernaert et al. [2013b].

Figure 7.5: Adding Note Attachments
Chapter 7. The Choose Editor in Practice

The biggest question marks in the case studies presented themselves how CHOOSE models should be composed. If all four viewpoint concepts are integrated into one model (e.g. Figure 6.10), the holistic overview is soon lost as the Practitioner goes more in detail and consequently the diagram increases in size.

A second approach is to create separate models for the different Viewpoints and then create relations between elements of different models through the properties view. Both options were explored in SMEs, where both proved to have advantages as well as disadvantages.

7.3.1 Separate Viewpoint Models

The first approach is to construct a model per CHOOSE Viewpoint. As such, the Practitioner could create separate Goal tree, Actor, Process and Object models (see Figure 7.6).

![Figure 7.6: Separate CHOOSE viewpoint models](image)

To create relations between the elements of different diagrams, the models need to be linked to each other:

*Right click the canvas→Load Resource→Browse Workspace→modelName.choose*

In doing so, relations between elements can be created, modified or deleted through the Properties View (Figure 7.7). As GMF does not support a search function, elements on the canvas cannot be located through a search-function. This implies that finding a specific element becomes difficult as the models are complex. In that regard, creating a separate model per CHOOSE Viewpoint can increase the manageability of the model, as less elements populate it.
When another model is loaded, shortcuts can also be imported. A shortcut is an element of a different model that is imported into the current model.

There are several restrictions concerning this way of modeling though. First, it requires the Practitioner to load each diagram in order to create relations between them. Second, the graphical overview is reduced as the Practitioner can only view an element’s relation with other elements through the Properties View.

Nonetheless, this approach was preferred in the governmental SME. It allowed them to create a holistic organogram, as well as create a limited Goal-tree, capturing the essence of the enterprise.

### 7.3.2 Integrating Viewpoint Models

A second approach is to blend the elements of the four Viewpoints into one model. The Tyrecenter model from Bernaert [2011] (Figure 7.8) and the Social Media model (Figure 6.10) are examples of this approach. The Practitioner has a more complete view as all the elements connected to a certain node can be visually observed. In order to create relations between elements, the Practitioner no longer needs to load other models. Creating relations in the Properties View as well as graphically linking two elements on the canvas with the Connector tool are possible.

The obvious disadvantage of this approach is that the overview can go lost as the model becomes more complex. Section 7.4 provides two functions which can help manage the model’s complexity, by filtering out elements and providing an automatic layout algorithm. The integrated model approach was preferred by the Brewery SME, which we will discuss further in the following section.
7.3.3 SME Case Illustration

To provide a concrete illustration of how CHOOSE modeling can benefit enterprises, a real-life example of the brewery SME enterprise is given.

AS-IS and TO-BE models were created by the author based on the information acquired from the company. The two models were modeled at an abstract level to keep the models simple while still providing a holistic overview. This was attained by modeling Operations at an abstract level and assigning the Performance of the Operations to Roles. Human Actors were purposefully not used. In a separate Organogram model, the roles were then assigned to Actors.
AS-IS

A simplified model of the Brewery AS-IS model is shown in Figure 7.9. The brewery is divided into two sites. Site 1 includes the Filtration, Filling barrels and Logistics Operations. Site 2 includes Brewing and Fermentation, Managing the Reception Room and Administration.

The Plant Manager of Site 1 is Responsible for Overseeing the Filtration, Filling Barrels and Logistics, and in doing so he Controls Oversee Software A, that serves as Input to the Oversee Site 1 Operation.

Plant Manager 2 is Responsible for another Overseeing Process that includes the Brewing and Fermentation Operation, and in doing so he Controls Oversee Software B, that serves as Input for the Oversee Site 2 Operation. Note these are two completely different Oversee Software Systems.

The Main Brewer is Accountable for Overseeing Site 1, Overseeing Site 2 and Brewing beer, which includes Operations from both Sites. The Main Brewer does this by Monitoring the Oversee Software Objects A and B.

Looking at this model, the SME had a clear view of the inefficiency of the Overseeing Process. If for instance the Main Brewer wants to be kept up to date in de production process, he had to extract data from the two different Oversee Software systems, which were located in separate buildings.
TO-BE

The model then was redesigned to provide a solution to this problem, illustrated in Figure 7.10. The two Overseeing Software systems are replaced by one Software system which provides the Input for the Overseeing Process of which the Head Brewer is Accountable. Passwords are implemented in the Software, which allows the Plant Managers to only access the Overseeing data that they have access to. As a result of this change in structure, one obsolete Software system is omitted and the Head Brewer would no longer have to switch between Software systems.

![Brewery TO-BE model](image)

These models were much appreciated by the SME, as they stated that even such a simple representation of their enterprise can provide a lot of insight into its structure. The author brought up the subject of Enterprise Architecture Principles. The redesign of the overseeing process could be the base of a principle stating that whenever possible, IT systems need to be standardized and reused throughout the organization. Such a principle would prove to be cheaper as redundant investments are prevented and economies of scale can be exploited. [Proper and Greefhorst, 2011]

They would like to have seen a manner to quantify the models, for example by describing Key Performance Indicators per goal. By creating several TO-BE models, the possibility would arise to see the impact of the different TO-BE models on the performance indicators.
7.4 Managing Complexity

When the size of the model increases, the Practitioner can lose overview. As Gronback [2009] states, a monitor of at least 30 inch is recommended for graphical modeling. The CHOOSE editor enables two functions to help maintain a clear overview: an automatic layout algorithm and filtering options.

7.4.1 Automatic Layout Algorithm

In Appendix B, the installation instructions for the KIELER plug-in are given. The KIELER project provides automatic layout algorithms, with properties that can be set to an interactive mode. [Duderstadt, 2011] It is activated by pressing the KIELER button in the toolbar. For instance, let’s assume the automatic layout algorithm is applied. the Practitioner does not like the outcome and repositions several nodes. The layout algorithm then detects these repositionings and learns how the Practitioner prefers to position the nodes. Applying the algorithm with the properties stated in Appendix B to the brewery’s AS-IS model (Figure 7.9) renders Figure 7.11 as result.

![Brewery AS-IS model with the KIELER layout algorithm](image)

Figure 7.11: Brewery AS-IS model with the KIELER layout algorithm
7.4.2 Filtering options

A second feature that helps in reducing a model’s complexity is filtering out elements. By right clicking a connection or node and selecting Filters, the choice is given to filter the selected elements or the Visual Type. If Human Actor is selected and the latter option is selected, all Human Actors will be filtered out of the diagram. The elements remain in the model repository though, so no information is lost. To return all elements back to sight, select Show All Hidden Parts in the Filter Options. This can be a useful feature to make a holistic model while still maintaining a model by reducing the amount of populated items.

7.5 Model Comparison

An important aspect of enterprise architecture is the ability to make a AS-IS / TO-BE analysis. [Schekkerman, 2004] GMF supports model comparison, but it is a rather underdeveloped function. Appendix B provides the installation instructions for the required plug-ins. Figure 7.12 shows a comparison made between the brewery AS-IS and TO-BE models previously discussed. The Comparison View lists all added and deleted nodes and edges, while the selected change is indicated in the model view below. The brewery SME would have liked to see a more developed comparison analysis though. Their main desire was the ability to assign key performance indicators to goals, after which they would create several TO-BE models. The capability should be present then to compare the different models in regard to their impact on the connected Goals.

![Model comparison AS-IS/TO-BE](image)

**Figure 7.12: Model comparison AS-IS/TO-BE**
Part III

Evaluation and Conclusion
Chapter 8

CHOOSE Editor Evaluation

In this chapter, the author evaluates the CHOOSE editor based on the requirements discussed in Chapter 4. The first section describes the editor’s performance of the various ISO-9126 quality characteristics. In the second section, the editor is evaluated on the performance of the criteria set by Bernaert et al. [2013a]. The assessments are based on the author’s evaluation of the editor and conclusions drawn from the case studies.

8.1 Software Quality Characteristics

Functionality, Usability and Efficiency characteristics are not discussed as they can are related to the criteria from Bernaert et al. [2013a], covered in the section 8.2.

Reliability

Reliability defines the capability of the system to maintain its service under defined conditions for defined periods of time. [ISO, 2001] It concerns the frequency of failure of the editor, recoverability of data after a software failure and the tolerance of software faults. In regards to this characteristic, following findings were assessed:

- In previous prototypes, the editor was installed in a workspace containing other projects. When running the diagram, errors of conflicting plug-ins often occurred. When the final editor version was installed in a new Eclipse installation, no more errors occurred.

- When a diagram model was abruptly terminated, unsaved data was lost.

- A stress test was performed on the system. As the Eclipse platform crashed, this resulted in the loss of unsaved data.

- If the .choose domain file is renamed, the corresponding diagram file becomes corrupted, and all visual information is lost.
Maintainability

While usability for the end user is very important, it is also necessary that a software product is maintainable for current and future developers. For instance, each year, a new version of Eclipse is introduced, which often implies a number of plug-ins will be incompatible. Maintainability concerns the analyzability, changeability and stability of the editor.

The analyzability is defined by the effort needed for diagnosis of the cause of failures. If an error occurs, a notification along with error details are given in the Error View. It is possible to view which plug-ins the error occurs. It is difficult to find the source of an error though, causing the Toolsmith to debug the project. This involves placing breakpoints in the source code and letting the application run step per step until it reaches a breakpoint where the error has occurred, which is a difficult and elongated method.

Table 8.1 shows the amount of classes and logical source lines of code that make up the CHOOSE editor. From a model-driven point of view, GMF-based editors lacks in changeability. If the Toolsmith were to make changes to the domain model, the entire development process would have to be redone. From a more programmatic point of view, it is dependent on the Toolsmith’s EMF and GMF knowledge. The CHOOSE editor is composed of 358 Java classes, containing over 27,000 logical source lines of code, which justifies the statement of Seehusen and Stolen [2011], saying that the most difficult aspect of modifying GMF-based editors often depends on finding the correct place to modify the code.

<table>
<thead>
<tr>
<th></th>
<th># Java Classes</th>
<th>Logical Source Lines of Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHOOSE Model Plug-in</td>
<td>34</td>
<td>2,142</td>
</tr>
<tr>
<td>CHOOSE.Edit Plug-in</td>
<td>16</td>
<td>590</td>
</tr>
<tr>
<td>CHOOSE.Editor Plug-in</td>
<td>4</td>
<td>966</td>
</tr>
<tr>
<td>CHOOSE.Diagram Plug-in</td>
<td>304</td>
<td>23,608</td>
</tr>
<tr>
<td>TOTAL</td>
<td>358</td>
<td>27,306</td>
</tr>
</tbody>
</table>
Portability

The portability refers to how well the software can adopt to changes to new specifications or operating environments. As the editor is an Eclipse plug-in, it will only be functional on a compatible Eclipse environment. The effort required to install the software (Appendix B) is fairly low, but prone to error as the correct plug-in versions need to be installed to avoid incompatibility errors.

8.2 Editor Requirements

8.2.1 EA Approach Requirements

Help control the complexity of the enterprise and its processes and systems.

The CHOOSE editor reduces complexity by restricting the practitioner to only a handful of elements to describe the enterprise. This was illustrated in section 7.3.3., where the brewery SME could reduce its complexity and attain their Goals by making high-level AS-IS and TO-BE models composed of Operations, Roles and Objects. The AS-IS model revealed redundant elements, while the TO-BE model gives a visual overview of how the organization’s structure changes.

Provide a holistic overview of the enterprise.

Holistic overviews can be created in a number of ways. A viewpoint-specific model can be created, e.g. a Goal-tree model of the entire enterprise, which can then be linked to other diagrams. The overview can also be created by making a model containing all four viewpoints. Features such as filtering out elements and an automatic layout algorithm to help manage the models complexity was discussed in chapter 7. It is up to the enterprise however to choose the scope, e.g. making a Goal-tree of the company as a whole or of a single division.

Facilitate the translation from corporate strategy to daily operations through objectives.

As the CHOOSE metamodel is based on Goal Refinements, the requirement regarding objectives is fulfilled. [Bernaert et al., 2013b]

Suitable to SMEs (this conforms to the SME-specific criteria).

This requirement is divided into the requirements of section 8.2.2.
Interaction of organizational components across different domains.

As CHOOSE is based on the essential dimensions from EA frameworks used for modeling enterprises, it is providing an enterprise overview. [Bernaert et al., 2013b]

8.2.2 SME-specific Requirements

Provide a time efficient manner to work on strategic issues.

As the metamodel is kept to a minimum, the practitioner can develop models more quickly. The governmental SME director was asked to create a small model in the CHOOSE editor as well as in Archi, a tool for creating ArchiMate models. As the CHOOSE editor restricted the director to 12 elements, as opposed to Archi’s 48 elements, the modeling process went considerably faster. The CHOOSE editor’s process of manually adding folders and CHOOSE models to then link them together one by one was not well received. Especially when observing the workings of Archi, the governmental SME liked the process of creating a project, upon which an entire hierarchy of folders and empty models that are already linked to each other is created. This functionality was then pursued by the author, but was not able to get it implemented.

Low IT knowledge barrier, ensuring people with limited IT skills can apply the approach.

The CHOOSE approach itself requires a very low IT barrier, as there is a restricted amount of elements to model with. The CHOOSE editor does require some IT knowledge though, particularly due the editor installment and the various functions the practitioner needs to learn. However, it is still assumed that the IT knowledge barrier is limited.

Easy to learn, small effort in getting familiar with the IS, low need for external help.

In one SME, the modeling happened at a quick pace as the practitioner had EA experience. Very few explanations were needed on how to operate the editor. In the other SME, the CEO needed to be steered by the author to stop him from going too much into detail on the aspects that he was most familiar with, and thus losing the overview. He was also not used to working with computers, and thus preferred to narrate all his knowledge while the author implemented it in the editor.

Capture knowledge in the firm regarding performance of tasks and how things are done.

This was achieved by modeling operations, processes and projects, though on a high, abstract level of operations. As one of the SMEs had already described its processes in
BPMN2, the demand rose to provide a capability to link or integrate BPMN files into the CHOOSE models. This is part of the future research, as the author explored the option to link files to CHOOSE models but lacked the programming skills to achieve this goal.

**Easy knowledge sharing across the firm.**

As the CEO was involved in modeling with the CHOOSE editor, all knowledge was available. Although the CEO is best involved in modeling the strategic aspect of the enterprise, it can be best for others to model different parts of the enterprise. E.g. the governmental company, had over 100 employees, and while the CEO had good knowledge of the strategic management, his knowledge of operation, object and actor viewpoint element was far less abundant.

**Low cost of entry and ensure benefits exceed the costs.**

Eclipse and the CHOOSE editor plug-in are open-source and publicly available, installation costs are non-existent. However, a large monitor is desired to ease the modeling process. As the ease of learning is very high, a low amount of time is lost learning to apply the CHOOSE approach and using the CHOOSE editor. The firsts SME benefited from making knowledge explicit in models while the second SME found true added value in the holistic model.
Chapter 9

Conclusion

The final chapter starts by declaring the research limitations that the author and the editor development were subjected to. Following, the main limitations and thus future extensions that should be implemented in the CHOOSE editor are discussed. Last, a final conclusion is given, summarizing the proceeds of this thesis project.

9.1 Research Limitations

As the developer of the CHOOSE editor had no previous experience with EMF, GMF and has no great programming knowledge, a great amount of time was spent discovering and learning how to use the Eclipse Platform and its various frameworks. The CHOOSE editor was developed making the most use of the GMF code generation capabilities as possible. To develop more elaborate editors adorned with additional functions though, EMF and GMF can be provide a great foundation on which to build an editor, but a high degree of customization would need to be performed. This in turn, as discussed in Chapter 4, requires a high level of EMF and GMF knowledge.

Furthermore, by the time this thesis project comes to an end, the CHOOSE approach research is still ongoing. Additional action research is planned to better meet the end-user’s requirements. Therefore, it is presumable that the CHOOSE metamodel will undergo changes, requiring modifications to the CHOOSE editor as well.
9.2 Future Extensions

The author has kept a critical point of view throughout this thesis project and recognizes areas of further improvement. A non-exhaustive list of future editor extensions is described, based on self-evaluation and feedback attained from the case studies. Unfortunately, each of these extensions is not supported through GMF code generation, which is the main reason these functionalities are not present in the editor. This implies implementing these functions through custom code, which was outside the capabilities of the author.

- The editor should be made more user-friendly, as the user-friendly interface is not enough. First, when the Practitioner creates a new CHOOSE model, a list of viewpoint folders with empty models should be generated by default, already linked to one another. Secondly, the editor should be deployed as a stand-alone executable file, omitting all its Eclipse dependencies.

- A much desired feature in the case studies was adding the capability of attaching or implementing other files into a CHOOSE model. In doing such, a CHOOSE model could serve as a holistic overview model containing user guides, excel production sheets, BPMN files...

- The possibilities to extract data from the CHOOSE model should be explored as well. For instance, by creating an Actor Viewpoint model and assigning them to Operations through RACI relations, it should be possible to extract a RACI performance matrix to e.g. Excel.

- Another desired feature attained from the case study is the quantification of the CHOOSE models. By adding key performance indicators to Goals, various strategic options can be compared according to their impact on the KPIs.

- Although there has already been performed a lot of action research towards the CHOOSE approach itself, CHOOSE needs to be further validated by introducing the tool support in SMEs.
9.3 Conclusion

In this thesis project, the author extensively tested and used the CHOOSE metamodel as the foundation on which to develop an Eclipse graphical editor plug-in. This plug-in serves as tool support for the CHOOSE approach. While developing the CHOOSE editor, the author made as much use of the GMF framework as possible. This framework enables the Toolsmith to automatically generate code in a model-driven approach. During the development phase, criteria to suit the target group such as user-friendliness were always kept in mind.

As a result of the authors restricted programming knowledge, many of the desired features a powerful enterprise architecture tool should possess are listed as future extension. Should these extensions be made possible, the CHOOSE editor would present a very potent tool, characterized by simplistic yet powerful features, user friendliness and a well-supported and validated metamodel. The editor tool has been tested and validated in two Belgian SMEs, after which the editor was modified according to the feedback received. The case studies proved successful, as they delivered added value to the SMEs while presenting no other cost than the time lost creating the CHOOSE models. The received feedback confirmed the need for additional features though.

Furthermore, this thesis has been very educational, by exploring the capabilities of Model-Driven-Architecture. It is a concept that can greatly reduce software development time while at the same time increase the quality of the code. Unfortunately, GMF is still a very error-prone framework, characterized by the lack of documentation and a high complexity of the generated code. Nonetheless, it is a powerful MDA framework, enabling a large base of users to create powerful graphical editors despite having limited IT skills.
Bibliography


Bibliography


Christopher Guntli. Create a dsl in eclipse. 2010.


MqE Iacob, H Jonkers, M Lankhorst, E Proper, and DAC Quartel. Archimate 2.0 specification. 2012.


ISO. Iso 8402, 1986.


AG Kleppe. A language description is more than a metamodel. 2007.


Astrid Van Acker. Literatuurstudie naar de bekendste raamwerken voor enterprise architectuur. 2012.


Appendix A

Summary of the CHOOSE concepts and Language Notation
### Appendix A. Summary of the CHOOSE concepts and Language Notation

#### Figure A.1: CHOOSE Node Concepts. Definitions from Bernaert et al. [2013b]

<table>
<thead>
<tr>
<th>CONCEPT</th>
<th>DEFINITION</th>
<th>VISUAL REPRESENTATION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Object Type</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Goal</td>
<td>An end state that an actor wishes to achieve and that is to be brought about or sustained through appropriate operations.</td>
<td>Goal</td>
</tr>
<tr>
<td>Refinement</td>
<td>Groups lower-level goals that all have to be fulfilled in order to fulfill a higher-level goal. Different refinements for one higher-level goal express different alternatives.</td>
<td></td>
</tr>
<tr>
<td>Actor</td>
<td>An organizational entity that is capable of performing operations.</td>
<td>Actor</td>
</tr>
<tr>
<td>Human Actor</td>
<td>A human being who is able to do performance of operations.</td>
<td>Human Actor</td>
</tr>
<tr>
<td>Role</td>
<td>The responsibility for the performance of specific operations, to which a human actor can be assigned who performs the role.</td>
<td>Role</td>
</tr>
<tr>
<td>Software Actor</td>
<td>A software system or part of a software system that encapsulates its behaviour and data to do performance of operations.</td>
<td>Software Actor</td>
</tr>
<tr>
<td>Device</td>
<td>A hardware resource or physical equipment that is able to do performance of operations.</td>
<td>Device</td>
</tr>
<tr>
<td>Operation</td>
<td>Internal behaviour that needs objects as input and produces objects as output. In order to operationalize goals, it can be a process or project.</td>
<td>Operation</td>
</tr>
<tr>
<td>Process</td>
<td>A behaviour element that groups behaviour based on an ordering of activities with the objective of carrying out work. It is intended to produce a defined set of products or business services.</td>
<td>Process</td>
</tr>
<tr>
<td>Project</td>
<td>A temporary endeavour undertaken to create a unique product, service or result.</td>
<td>Project</td>
</tr>
<tr>
<td><strong>Object Type</strong></td>
<td>A passive element that has relevance from a business, information, or technological perspective. It corresponds to a real world counterpart that may or may not be physical.</td>
<td>Object</td>
</tr>
</tbody>
</table>

---

*Figure A.1: CHOOSE Node Concepts. Definitions from Bernaert et al. [2013b]*
# Appendix A. Summary of the CHOOSE concepts and Language Notation

Table: CHOOSE Relation Concepts

<table>
<thead>
<tr>
<th>Concept</th>
<th>Definition</th>
<th>Visual Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Connection Type</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OR-Ref</td>
<td>Refines a goal in alternative refinements.</td>
<td></td>
</tr>
<tr>
<td>AND-Ref</td>
<td>Expresses that a goal can be satisfied by satisfying all its sub-goals.</td>
<td></td>
</tr>
<tr>
<td>Conflict</td>
<td>Interconnects goals to capture potential conflicts among them.</td>
<td></td>
</tr>
<tr>
<td>Wish</td>
<td>Captures the fact that an actor would like a goal to be satisfied.</td>
<td></td>
</tr>
<tr>
<td>Assignment</td>
<td>An actor is assigned to a goal if it is required to restrict its behaviour so as to satisfy the goal.</td>
<td></td>
</tr>
<tr>
<td>Operationalization</td>
<td>Refers to the process of mapping goals (ends) to operations (means) realizing them.</td>
<td></td>
</tr>
<tr>
<td>Concern</td>
<td>Connects goals to the objects to which they refer.</td>
<td></td>
</tr>
<tr>
<td>Division</td>
<td>Indicates that an actor groups a number of other actors.</td>
<td></td>
</tr>
<tr>
<td>Supervision</td>
<td>A supervises reports to a supervisor.</td>
<td></td>
</tr>
<tr>
<td>Performs</td>
<td>Links roles with human actors that fulfill them</td>
<td></td>
</tr>
<tr>
<td>Responsible</td>
<td>The actor who must perform the work, is responsible for the activity until the work is finished and approved by an accountable.</td>
<td></td>
</tr>
<tr>
<td>Accountable</td>
<td>The actor who must approve the work performed by the actor responsible for an operation.</td>
<td></td>
</tr>
<tr>
<td>Consult</td>
<td>The actors whose opinion is sought while performing the work, and with whom there is two-way communication.</td>
<td></td>
</tr>
<tr>
<td>Inform</td>
<td>The actor who is kept up-to-date about the progress of an activity and/or the results of the work, and with whom there is just one-way communication.</td>
<td></td>
</tr>
<tr>
<td>Monitoring</td>
<td>An actor monitors an object if it can see values of the object, without changing them.</td>
<td></td>
</tr>
<tr>
<td>Control</td>
<td>An actor controls an object if it can create an object or change the values of the object.</td>
<td></td>
</tr>
<tr>
<td>Includes</td>
<td>Groups sub-operations in the super-operations of which they are part.</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>Designates an object to which the operation applies.</td>
<td></td>
</tr>
<tr>
<td>Output</td>
<td>Designates an object on which the operation acts.</td>
<td></td>
</tr>
<tr>
<td>Association</td>
<td>Models a relationship between objects that is not covered by another, more specific relationship.</td>
<td></td>
</tr>
<tr>
<td>Aggregation</td>
<td>Indicates that an object groups a number of other objects.</td>
<td></td>
</tr>
<tr>
<td>Specialization</td>
<td>Indicates that an object is a specialization of another object.</td>
<td></td>
</tr>
</tbody>
</table>

Figure A.2: CHOOSE Relation Concepts. Definitions from Bernaert et al. [2013b]
Appendix B

Installation and Source Code of Editor

B.1 Setup

As updates and new versions of the used plug-ins are often introduced which can cause incompatibility issues, the correct Eclipse version and plug-in builds needs to be installed. The exported version of the CHOOSE editor is developed with Eclipse Juno (2012). It is recommended to create a new Eclipse installation with the following configuration:

- Eclipse Juno SDK 4.2.2 (M2 or higher)
- Ecore Tools SDK 1.1.0
- Eclipse EGit 2.3.1
- EMF SDK 2.8.3
- EMF Compare Core SDK 2.1.0
- EMF Compare EcoreTools Integration 2.1.0
- EMF Compare GMF Integration 2.1.0
- EMF Compare UML2 Integration 2.1.0
- GMF Notation SDK 1.6.0
- GMF Runtime SDK 1.6.2
- GMF Tooling SDK 3.0.2
- KIELER Editor for Graphs 0.6.0
Appendix B. Installation and Source Code of Editor

- KIELER Layout for GMF 0.8.0

Start with installing the Eclipse Classic 4.2.2 SDK from the Eclipse download page\(^1\). Unzip the file in a new folder and start the Eclipse installation. To enable the Properties View: select Window→Show View→Other→Properties

**Go to Help→Install New Software→Work With:**

1. [http://download.eclipse.org/releases/juno](http://download.eclipse.org/releases/juno)
   
   Modeling→Ecore Tools SDK, EMF SDK, GMF Notation SDK, GMF Runtime SDK, GMF Tooling SDK

   
   EMF Compare→EMF Compare SDK
   
   EMF Compare Tools Integration→EMF Compare EcoreTools Integration, EMF Compare GMF Integration, EMF Compare UML2 Integration

   Add the CHOOSE models to the model comparison by going to the Eclipse workspace: Window→Preferences→General→Content Types→Add .choose and .choose_diagram to both the Ecore Tools diagram for compare and EMF Compare.

3. [http://rtsys.informatik.uni-kiel.de/~kieler/updatesite/](http://rtsys.informatik.uni-kiel.de/~kieler/updatesite/)
   
   KIELER 0.8.0→KIELER Editor for Graphs, KIELER layout for GMF

   Through self-evaluation, the best results for the automatic layout algorithm have been found with the following properties (specified in the Eclipse Layout View):

   - Crossing Minimization = INTERACTIVE
   - Cycle Breaking = INTERACTIVE
   - Direction = DOWN
   - Edge Routing = ORTHOGONAL
   - Separating Connected = true

Appendix B. Installation and Source Code of Editor


Eclipse Git Team Provider → Eclipse EGit

B.2 Shared Environment

The CHOOSE editor plug-in and source code are publicly available for download at https://github.com/SimonZutterman

B.3 Installation of the CHOOSE editor

- Go to https://github.com/SimonZutterman/CHOOSE-Plug-in and download the CHOOSE Eclipse Plug-in.zip file
- Open your installation of Eclipse Juno
- Select Help → Install New Software
- Select Add, then Archive, and locate the Choose Eclipse Plug-in.zip on your system.
- Select the editor and follow instructions until the editor is installed.
- Restart Eclipse when asked.

B.4 Source Code of the Editor

The source code is also available, which can be imported to look into the code or look at the EMF and GMF models. This can be either imported via Git or imported manually

GIT

- Go to https://github.com/SimonZutterman/CHOOSE-Source-Code
- Copy the Git Read-Only link to your clipboard
- In Eclipse, Select File → Import → Project from Git
- Enter the copied link
- Import the source code files into your workspace
Manually

- Go to https://github.com/SimonZutterman/CHOOSE-Source-Code
- Download the source code files as .zip
- Unpack the .zip file in your system
- In Eclipse, Select File→Import from existing project
- Locate the source code files on your system
- Follow the import instructions, copying the files to your workspace
Appendix C

Eclipse GMF hints

This section gives the reader a few tricks for customizing code in the GMF framework. Four useful features are given, which are also used in the CHOOSE editor.

Multiline Text Labels

Multiline labels are not supported by default by GMF. Especially when models have to be built containing a lot of elements, it is useful to save space using multiline labels. To attain this feature, two java files need to be adjusted. The first is the label edit part, the second the edit part of the node containing the label. To illustrate this, we will provide multilink capabilities to the Actor’s Name label.

1) locate the ActorNameEditPart (found in Choose.diagram/src/chooseeditor/diagram/edit/parts/ActorNameEditPart.java)

Locate the code in the frame below.

```java
/**
 * @generated
 *
 * protected DirectEditManager getManager() {
 *     if (manager == null) {
 *         setManager(new TextDirectEditManager2(this, null,
 *             DefaultNameEditPartFactory.getTextCellEditorLocator(this)));
 *     }
 *     return manager;
 * }
 */
```

…and replace it with:

```java
/**
 * @generated NOT
 */
```
Appendix C. Eclipse GMF hints

```java
protected DirectEditManager getManager() {
    if (manager == null) {
        setManager(new TextDirectEditManager2(this, WrapTextCellEditor.class, ChooseEditorEditPartFactory.getTextCellEditorLocator(this));
    }
    return manager;
}
```

2) go to the ActorEditPart and set the TextWrap for the label to (true)

```java
/**
   * @generated NOT
   */
private void createContents() {
    fFigureActorNameLabel = new WrappingLabel();
    fFigureActorNameLabel.setText("<...")
    fFigureActorNameLabel.setTextWrap(true);
    fFigureActorNameLabel.setFont(FFIGUREACTORMENAMELABEL_FONT);
}
```

Custom Property View

By default, a Node’s Properties view is arranged alphabetically, and is not customizable through the GMF development.

There is a good explanation to be found on the eclipse forums to solve this issue:


Gradient Colored Nodes

To add a gradient node color, simply add the following code to the node’s inner class:

```java
public class ActorFigure extends RoundedRectangle {
    /**
       * @generated NOT
       */
    protected void fillShape(Graphics graphics) {
        // Backup the graphics colors
        Color bgColor = graphics.getBackgroundColor();
        Color fgColor = graphics.getForegroundColor();
        // Set the graphics color
```
Middle Decorations

To allow decorations being displayed in the middle, implement the following code and import all missing packages:

```java
/**
 * @generated
 */
public class ActorAssignmentFigure extends PolylineConnectionEx {
    /**
     * @generated NOT
     */
    public ActorAssignmentFigure() {
        this.setForegroundColor(ColorConstants.darkGray);
        setTargetDecoration(createTargetDecoration(), new ArrowLocator(this, ←
            ConnectionLocator.MIDDLE));
    }
}
```