Tool Support for Enterprise Architecture Analysis

with application in cyber security

MARKUS BUSCHLE

Doctoral Thesis
Stockholm, Sweden 2014

TRITA-EE 2014:025
ISSN 1653-5146
ISRN KTH/ICS/R-14/03-SE
ISBN 978-91-7595-159-1

Industrial Information and Control Systems
KTH, Royal Institute of Technology
Stockholm, Sweden

Akademisk avhandling som med tillstånd av Kungl Tekniska högskolan framlägges till offentlig granskning för avläggande av Doctor of Philosophy June 11, 2014 i F3, Kungl Tekniska högskolan, Lindstedsvägen 26, Stockholm.

Tryck: Set in LaTeX by the author
Cover illustration by Evelina Ericsson
Printed by Universitetsservice US AB

für Wolfgang Buschle

Abstract

In today's companies, business processes and information technology are interwoven. Old and new systems as well as off-the-shelf products and tailored solutions are used. This results in heterogeneous, often complex IT landscapes. The impact of changes and the affected systems are difficult to identify. However, volatile business environments and changing customer requests require organizations to adapt quickly and to frequently make decisions about the modifications of their information technology.

IT management aims at generating value from the usage of information technology. One frequently used IT management approach is Enterprise Architecture. Company-wide models are used to obtain a holistic picture. These models are usually created using Enterprise Architecture modeling tools. These tools frequently have strong documentation capabilities. However, they often lack advanced analysis functionality. Specifically, such tools do not offer sufficient support for the analysis of system properties, such as cyber security, availability or interoperability. The ability to analyze a set of possible scenarios and predict the properties of the modeled systems would be valuable for decision-making. Changes or extensions could be evaluated before their implementation. In other domains, for example, in architecture in its classical meaning or in the development of machines, the analysis of models is a common practice. Typically, CAD tools are used to perform analysis and support decision-making. It is thereby possible to investigate the stability of buildings or the performance of engines without the need for empirical testing.

The contribution of the research work documented in this thesis is a software tool with a particular focus on the analysis of Enterprise Architecture models and thereby support for decision-making. This tool combines state-of-the-art Enterprise Architecture tooling with advanced analysis capabilities that, until now, were only offered by modeling tools for other domains. The presented tool possesses two components. One component allows the creation of a metamodel capturing Enterprise Architecture analysis theory, for example, relevant concepts in the context of cyber security and how they relate to each other. The other component supports the instantiation of the metamodel into an Enterprise Architecture model. Once a model is in place, it can be analyzed with regards to the previously specified theory so that, for instance, a cyber security evaluation can be conducted.

The analysis tool was partly developed within the context of a larger research project on cyber security analysis. However, the tool is not restricted to applications within this field. It can be used for the evaluation of numerous system properties. Several authors contributed to the tool both on an implementation level and in the development and design of the tool's features. The performed research followed the Design Science methodology. First, the objectives of a tool for Enterprise Architecture analysis were defined. Next, an artifact was designed and developed in terms of a software tool. This tool was then demonstrated and evaluated against the objectives. Lastly, the results were communicated to both academic and non-academic audiences.
Keywords: Enterprise Architecture, Decision-making, Model-based analysis, Property analysis, Cyber security, Software tool, Design Science

Sammanfattning

På de flesta företag är affärsprocesser och IT tätt sammanvävda, det förekommer en kombination av gamla och nya system, så väl standardiserade produkter som skräddarsydda lösningar används. Detta resulterar i heterogena och ofta komplexa IT-landskap. Effekten av förändringar och vilka system som påverkas är ofta svårt att identifiera. Trots det kräver dagens ostadiga företagsmiljöer och kundkrav att organisationer snabbt anpassar sig till, och kontinuerligt fattar beslut om ändringar av deras informationsteknologi.

IT- management syftar till att generera värde ur användandet av informationsteknik. En vanlig IT- management strategi är Enterprise Architecture (organisationsövergripande arkitektur) som rekommenderar skapandet av företagsövergripande modeller för att få en helhetsbild av en verksamhet. Vanligtvis skapas dessa modeller med hjälp av modelleringsverktyg anpassade för att dokumentera en verksamhets nuläge. Avancerade analysfunktioner saknas ofta i befintliga verktyg, som därför ger svagt stöd vid utvädring av specifika systemegenskaper som till exempel IT-säkerhet, tillgänglighet eller interoperabilitet. Förmågan att analysera möjliga scenarier och förutsäga de modellerade systemens egenskaper skulle vara värdefullt för beslutsfattare. Förändringar skulle på så sätt kunna utvärderas före implementering i verksamheten. Inom andra områden, till exempel arkitektur i dess klassiska mening, eller vid utveckling av maskiner, är analys av modeller ett vanligt tillvägagångssätt. CAD verktyg används ofta för att utföra analyser och som stöd vid beslutsfattande. Därigenom är det möjligt att utvärdera byggnaders stabilitet eller motorers prestanda utan omfattande tester eller mätningar.

Bidraget från forskningen i denna avhandling är ett verktyg för analys av Enterprise Architecture modeller och därmed stöd för beslutsfattande. Detta verktyg kombinerar moderna Enterprise Architecture verktyg med avancerade analysfunktioner som fram till idag endast applicerats inom andra domäner. Analysverktyget har två komponenter,en komponent som gör det möjligt att beskriva Enterprise Architecture analysteori och hur de relaterar till varandra, exempelvis inom IT-säkerhet. En annan komponent stödjer användandet av denna teori i en Enterprise Architecture-modell. När en arkitekturmodell är på plats kan verktyget hjälpa användare att utföra analys med avseende på tidigare angiven teori, så att till exempel en utvärdering av IT-säkerhet kan genomföras.

Analysverktyget har delvis utvecklats inom ramen för ett större forskningsprojekt om IT-säkerhetsanalys. Verktyget är dock inte begränsat till tillämpningar inom detta område utan kan även användas för att utvärdera andra systemegenskaper. Flera personer har bidragit till verktyget både avseende design, utveckling och implementering av verktygets funktioner. Forskningen har genomförts enligt Design Science-metodiken. Först definierades krav på ett verktyg för Enterprise Architecture analys. Därefter utformades och utvecklades ett verktyg. Detta verktyg har sedan demonstreras och utvärderas i förhållande till kraven. Resultaten har slutligen presenterats i både akademiska och icke-akademiska forum.

Nyckelord: Enterprise Architecture, beslutsfattande, modellbaserad analys, egenskapsanalys, cybersäkerhet, mjukvaruverktyg, Design Science

Zusammenfassung

In heutigen Unternehmen sind Geschäftsprozesse und Informationstechnologie untrennbar miteinander verwoben. Alte und neue Systeme, Standardprodukte und maßgeschneiderte Technologien finden dabei Verwendung. Dies resultiert in heterogenen komplexen IT-Landschaften. Die Auswirkungen von Änderungen auf betroffene Systeme sind schwer zu erkennen. Dynamische wirtschaftliche Rahmenbedingungen und wechselnde Kundenanforderungen erfordern schnelle Anpassung und Entscheidung über Änderungen der Informationstechnologie.

IT-Management hat als Ziel Wertschöpfung aus der Nutzung von Informationstechnologie zu fördern. Ein gängiger Ansatz ist dabei Enterprise Architecture. Durch unternehmensweite Modelle erhält man ein ganzheitliches Bild. Diese Modelle werden mittels Enterprise Architecture Modellierungswerkzeugen erstellt. Die Werkzeuge haben häufig ausgeprägte Dokumentationsfunktionen, aber keine tiefgreifende Analysefunktionalität. Insbesondere unterstützen die Werkzeuge die Analyse von Systemeigenschaften, wie z.B. Cyber-Security, Verfügbarkeit und Interoperabilität nur unzureichend. Die Analyse potentieller Szenarien und die Vorhersage ihrer Eigenschaften fördert die Entscheidungsfindung. Änderungen oder Erweiterungen könnten so vor ihrer Umsetzung modelliert werden. In anderen Bereichen, z.B. in der Architektur im klassischen Sinne oder bei der Entwicklung von Maschinen, ist die Analyse von Modellen üblich. Typischerweise werden dabei CAD-Werkzeuge verwendet. Dadurch ist es möglich, die Stabilität von Gebäuden oder Leistung von Motoren zu untersuchen ohne empirische Tests durchzuführen.

In dieser Arbeit wird ein Softwarewerkzeug zur Analyse von Enterprise Architecture Modellen vorgestellt. Es kombiniert Funktionalität von aktuellen Enterprise Architecture Werkzeugen mit erweiterter Analysefähigkeit, die es bis heute nur bei Modellierungswerkzeugen in anderen Domänen gibt. Das vorgestellte Analysewerkzeug besitzt zwei Komponenten. Eine Komponente dient der Theoriespezifikation für die Analyse von Enterprise Architecture-Modellen. Relevante Konzepte und deren Beziehung zueinander können definiert werden. Die zweite Werkzeugkomponente unterstützt die Instanziierung der Analysetheorie in Architekturmodellen. Die Modelle können anschließend unter Berücksichtigung der zuvor festgelegten Theorie (die z.B. eine Cyber-Security Bewertung beschreibt) ausgewertet werden.

Das beschriebene Analysewerkzeug wurde teilweise im Rahmen eines umfangreicheren Forschungsprojekts zur Cyber-Security Analyse entwickelt. Es ist jedoch nicht auf diesen Kontext beschränkt, sondern kann zur Analyse einer Vielzahl von Systemeigenschaften benutzt werden. Eine Anzahl von Autoren haben zur Implementierung sowie zur Entwicklung und Gestaltung der Funktionalität beigetragen. Das Werkzeug wurde gemäß dem Design Science Ansatz entwickelt. Zu Beginn wurden Anforderungen an ein Werkzeug für die Enterprise Architecture- Analyse ermittelt. Danach wurde das Werkzeug designed, implementiert und anhand der Anforderungen ausgewertet. Abschließend wurden potentielle Anwender über das Werkzeug informiert.

Stichworte: Enterprise Architecture, Beschlussfassung, Modellbasierte Analyse, Eigenschaftsanalyse, Cyber-Security, Softwarewerkzeug, Design Science

Acknowledgments

Many people have supported me during my Ph.D. studies. I owe a debt of gratitude to my supervisors Pontus Johnson, Mathias Ekstedt and Göran Ericsson.

Torsten Cegrell was not only kind enough to hire me, but, together with Judy Westerlund, created a unique, international, inspiring and rewarding working environment. Thank you both. You two really helped me get started in Sweden and made me feel at home.

Being a Ph.D. student sometimes has its ups and downs. I was lucky to have supportive colleagues ensuring that the good times outweighed the rough moments. In particular, I want to thank Joakim Lilliesköld, Robert Lagerström, Per Närman, Pia Närman, Teodor Sommestad, Johan Ullberg, Moustafa Chenine, Ulrik Franke, David Höök, Johan König, Waldo Rocha Flores, Liv Gingnell, Nicholas Honeth, Claes Sandels, Matus Korman and Margus Välja.

I especially want to thank Khurram Shahzad, who contributed to my research probably more than anyone else.

Four colleagues in particular made my life more enjoyable. A big thank you goes to Kun Zhu, who shares my passion for traveling and photography as well as my curiosity and interest in trying out Stockholm's restaurants.

I also want to thank Hannes Holm, who inspired me both in numerous work-related conversations and at the gym. Frequently, both things happened at the same time.

Annica Johannesson deserves acknowledgment for doing the magic behind the scenes. I always enjoyed working with you. This thesis would not be the same without your support!

Ett stort tack goes to Evelina Ericsson, who over the last few years became a very close friend of mine. Thank you for teaching me Swedish and introducing me to Sweden. I cannot imagine what the last five years would have been like without your continuous help!

I am grateful to my international co-authors Oliver Holschke, Jannis Rake-Revelant, Dick Quartel, Florian Matthes, Sabine Buckl, Chrisitan M. Schweda, Sascha Roth, Matheus Hauder, Sebastian Grunow, Sasi K. K. and Nithin Somasundaran.

Furthermore, I would like to thank Daniel Feller, Torsten Derlat and Marten Schönherr. Without you, I would never have ended up in Stockholm.

While writing this thesis, I was supported by John and Teri Schmelzel. Thanks a lot!

Without a doubt, Rafaela Buschle is the person I need to thank the most. You helped me in uncountable ways and were always there for me whenever I got lost. Thank you! I will do the same for you when it is time for your Ph.D. studies.

I am grateful to my friends who supported me during the last several years: Sebastian Wrede, Virginia Hüntemann, Baki Cakici, Hanna Sjögren, Christiana Gransow, Andrea Bobrowsky, Andrea Feller, Anna Önnhage, Yunle Mo and Maria Zayas as well as my siblings from another mother (and father): Erin Schmelzel, Anne Copple and Adam Schmelzel.

I would also like to thank Ursula Buschle and Eberhard Riedel for their support.

Thank you all!

Stockholm, June 2014
Markus Buschle

Table of contents

List of Figures

List of Tables

Chapter 1

Introduction

This first chapter of this thesis introduces the topic of the documented research. The first section explains the motivation for the research performed. Furthermore, the purpose of the presented research is stated. In the second section, the main contributions of the presented research work are summarized. Thereafter, the research work documented in this thesis is delimited, including a clarification of its relationship to the work of other authors. The collaborative research project to which the presented research partly contributed is thereafter introduced. Lastly, the fourth section introduces to the structure of the remainder of this thesis. This thesis targets several audiences. First and foremost, academic audiences will be informed of the author's research and findings. Second, the result of the research documented in this thesis is a software tool that can support the work of practitioners. Therefore, professionals and potential tool users are addressed to inform them about the tool and convince them of the value of its use.

1.1 Research motivation

The business of contemporary enterprises now depends more than ever on the usage of IT (information technology). The advent of the Internet; the mass production of inexpensive standard IT components, such as desktop computers, servers and network equipment; and the achievements made to simplify software development are the main drivers of this trend. Recent phenomena such as cloud computing, the ubiquitous usage of IT and software as services and the penetration of smartphones in the Western and Eastern civilizations connect the business area even more closely to the IT domain. This is an observable trend that is not limited to a particular business area or part of our society. Instead, it is almost impossible to find companies not using any IT systems at all. Considering many of the successful IT companies that were founded during the last two decades, it becomes apparent that those companies tend to have business models that are strongly connected to the usage of IT. Deloitte, as part of their 2013 Technology Fast 500™ ranking,

found that "New technologies like cloud and software as a service (SaaS) are at the forefront of the exponential growth we are seeing in software companies"[60]. Five of the seven fastest-growing IT companies listed in the Fortune 100 ranking of the fastest-growing IT companies are companies whose business model is to offer various IT-based services [72].

However, the extensive usage of IT is not a unique feature of newly established enterprises. Even fairly old areas, such as the finance sector, the insurance industry, the electric power industry and the defense industry, now rely on the usage of IT. Companies operating in these domains use information technology to automate tasks whose manual performance would be expensive. Decisions and transactions can be made much more quickly when performed digitally, and the outcome is often easier to predict. Storing data digitally saves space and makes it easy to retrieve information. The usage of IT as a means of communication allows the creation of virtual organizations with sites spread all over the world. Work can be performed collaboratively without the need for the physical presence of the team members and the traveling that goes along with it.

However, the sheer extent of usage of IT is by no means a quality assurance [48]. IT has become a commodity and, as every enterprise is using information technology, it is not possible to gain a competitive advantage by simply buying computer systems. The question is how to utilize information systems in an optimal way. This typically translates into the question of how to use IT in a way that is as cost efficient as possible and as closely aligned to the business goals of the company. Companies have to adapt quickly to satisfy their costumers and keep up with and overtrump their competitors. They need to be able to quickly offer new and interesting products to attract new customers and bind their existing ones. This requires companies to have flexible business processes that can be adjusted on demand.

As mentioned earlier, business and IT are typically connected. Therefore, modifications of a business process or the development of a new product are likely to propagate. Adjusting the business to meet the customers' demands also poses requirements on the underlying information technology. IT components need to be capable of supporting the current products and business processes, as well as future modifications, variations and preferably even completely new products.

Supporting their business by adding more and more IT components, many companies have aggregated a zoo of IT systems. Very often, one can find a heterogeneous setup. Many companies use hardware and software from different vendors in parallel, run different operating systems on their machines and use different software versions simultaneously. They utilize several integration technologies parallel to each other to connect their systems, consume IT services delivered by numerous providers and use different types of databases side by side. Companies use hardware that they bought several decades ago in combination with the latest technology. In addition, they combine tailored solutions developed to solve a specific problem with off-the-shelf products one can find almost everywhere. According to Garner, 355.2 million computer were sold in 2011, 74 % of which were procured for business pur-

poses [32]. Some large companies have several hundred thousand computer systems in use [117]. Almost 15 years ago, in 2000, the American Department of Defense had 10,000 computer systems consisting of 1.5 million computers [59]. These figures are likely to increase every year.

Administrating and coordinating these huge quantities of information systems cannot be achieved without using elaborate proceedings and methods. Already, it is common for small companies to have a dedicated role dealing with IT-related questions. Large companies have large departments working exclusively with information systems.

The discipline of information technology management aims to help enterprises in their attempts to steer their computer systems. In particular, it provides a toolkit for planning and directing the evolutionary development of the (IT) landscape [107]. The overall goal of IT management is to generate value from the usage of technology. This can only be achieved if business strategies and technology are aligned. Thus, it must be stressed that IT management is more than just the management of IT systems. Topics other than business IT alignment that are typically included in IT management include governance, strategic planning, financial management, risk analysis and organizational performance [91].

Enterprise Architecture (EA) is one approach to IT management, wherein models, e.g., diagrammatic illustrations or textual descriptions, are used to represent enterprises holistically. Enterprise Architecture models include both the business domain and the IT area found in contemporary companies. By connecting these two domains, Enterprise Architecture models foster communication between different involved stakeholders. In this way, Enterprise Architecture supports business IT alignment.

Moreover, Enterprise Architecture models can be used to describe scenarios. This includes illustrations of how the company looks today (as-is models) as well as potential future setups (to-be models) [153]. By supporting the comparison of different alternative scenarios, Enterprise Architecture also provides decision support. Stakeholders can evaluate models to recognize the strengths and weaknesses of a particular scenario. This helps to identify a desired future scenario. Additionally, the as-is model can be compared with the future goal to identify the necessary changes and milestones while transitioning[239].

Enterprise Architecture models are typically created based on metamodels that provide syntax and semantics on how to create the descriptions. These metamodels, as well as methods detailing what and how to model, are typically called Enterprise Architecture frameworks. Over the years, a number of public and private organizations have developed Enterprise Architecture frameworks, including the Zachman framework [276], TOGAF[265] and DODAF [98].

The usage of Enterprise Architecture to describe the relationship between the business and IT domains typically results in fairly large models. The description of a small subset of the considered organization can already lead to a model with several hundred elements. If a companywide model is to be created, depending on the size of the considered company, one can expect several hundred thousand elements

to be part of the description. The manual creation of models of this size is both expensive and prone to errors. Instead, people working in IT management typically perform this task using Enterprise Architecture modeling tools. These tools allow the collaborative creation of models as well as the visualization of specific aspects depending on the target audience. Models created with Enterprise Architecture tools typically document the current state of the enterprise. Once a model is in place, a process owner can, for example, identify the IT systems that are used to execute a certain process. However, the sheer usage of Enterprise Architecture tools is not sufficient to ensure that business strategies and IT capabilities are sufficiently adjusted. Gartner states that "Enterprise architecture tools can provide tremendous business value, but only when aligned with the needs of the organization. Enterprise Architecture tools must be selected, deployed and managed carefully to ensure proper ROI.[87]"

Information technology advisories such as Gartner [88]and academic groups such as the Department of Software Engineering for Business Information Systems at Technische Universität München (TUM) [166] evaluate Enterprise Architecture tools. They conduct these evaluations in collaboration with tool vendors and especially end-user customers. Furthermore, they use such criteria as ability to create visualizations, support of large-scale data, communication and collaboration support and usability to evaluate the strengths and weaknesses of those tools [166]. Both studies identify that the currently available tools are strong in terms of the creation of Enterprise Architecture models and the support of collaborative model creation.

However, current Enterprise Architecture tools often do not allow the investigation of the details of the described systems. In particular, Enterprise Architecture tools generally possess limited analysis capabilities with regards to system properties, such as cyber security, availability or interoperability. Answering such questions as "is this business process available, even under high pressure and many executions, given the company's infrastructure?" or "is this system likely to be target of a cyber security attack?" is not possible by those tools. One cause of this weakness is that Enterprise Architecture tools typically do not consider the attributes of the modeled elements completely way and, in particular, disregard how attributes impact each other. Doing so, they lack the ability to evaluate whether the scenario described in a certain model fulfills the requirements posed against it. This, however, is necessary to identify whether the description is preferable and, as a second step, whether the described scenario should be implemented or disregarded.

Gartner and TUM identify the need for Enterprise Architecture tools to "provide valuable information and analysis capabilities for strategic decision making"[87]. Evaluating 14 (Gartner) or 12 (TUM) Enterprise Architecture tools, both tool surveys identify that the analysis capabilities are not yet completely mature. In the Hype Cycle for Enterprise Architecture [90], 2013, it is stated that Enterprise Architecture tools are two to five years away from "capturing vital enterprise context background, along with content development and analysis capabilities across the

business, information, technology and solution architectures." On the other hand, TUM [166] notes that, "concluding this non-exhaustive list of ideas for possible topics in Enterprise Architecture management and development ... we regard mechanisms for performing simulations on the Enterprise Architecture or subsets thereof being a promising approach. With these simulations techniques complemented by methods for quantifying certain properties of the Enterprise Architecture, likewise metrics, we see the dawn of a new Enterprise Architecture management maturity level". Outside the domain of Enterprise Architecture, tools that combine modeling and analysis exist.

Numerous tools allow the investigation of subparts of organizations. Tools such as Opnet[176] focus on the performance analysis of computer networks and applications. In this tool, one first creates a visual representation of the infrastructure that should be investigated. Thereafter, this description can be used to investigate whether the applications of a considered company are sufficiently supported by the underlying information systems and networks. Other modeling tools focus on the description of a company's business processes. The ARIS Business Process Analysis Platform [3] allows the description of how different roles perform the activities that can be found at an organization. Additionally, how activities relate to each other and together form business processes can be modeled. It is even possible to simulate these processes to identify bottlenecks, overcapacity and suboptimal utilization of resources or redundant tasks.

Using MulVAL [206], one can model possible attacks on IT architecture. These models are created based on vulnerability scanners. NetSPA[11] and its successors GARNET [270] and NAVIGATOR[52] follow the same approach to support model-based vulnerability assessments. k-Zero Day Safety [266] is a tool for modeling zero-day attacks.

The ATHENA Interoperability Framework [21] includes a tool for the interoperability analysis of IT systems. The Analytical Availability Assessment of IT Services [163] allows the modeling and evaluation of service availability.

However, no available tool allows company-wide analysis covering both business and IT elements comprehensively. Unlike the available Enterprise Architecture tools, the available analysis tools do not focus on providing enterprise-wide decision support. Instead, the analysis tools focus on limited parts of a considered organization. Moreover, the discussed tools only allow the consideration of one or a few system properties [1] . An analysis of numerous system properties simultaneously is typically not supported by these tools. In particular, no tradeoffs between the system properties from different domains, such as tradeoffs between cyber security and organizational structure, are possible.

Switching focus from enterprises to the products they create and sell, the design technology CAD (computer-aided design) [63] is often encountered. This approach is commonly used in the making of machines, vehicles and buildings, i.e., models

[1] The term "system" is here used in its wider meaning, i.e., "a complex whole" [207], and is not limited to IT systems.

of the artifacts that will be created. These models provide a great benefit: it is easy to perform calculations of how the artifacts would behave instead of testing them empirically. Empirical testing, such as crash tests, is expensive and time consuming. Based on CAD calculations, the optimal material for a given purpose can be identified or an optimal setup for a certain construction can be selected.

Current Enterprise Architecture tools are comparable to available CAD tools without the ability to analyze the design, i.e., without the functionality to simulate crash tests, calculate the stability of buildings or investigate the performance of engines. The available Enterprise Architecture tools generally only support the creation of descriptive models and lack advanced analysis capabilities [166]. Investigations of system availability or how well an organization is capable of fulfilling its goals are generally not possible, nor is it possible to analyze cyber security aspects to identify vulnerabilities.

The research documented in this thesis was intended to completely fulfill the CAD analogy.

> **The purpose of the described research is therefore to develop and demonstrate an Enterprise Architecture modeling tool with a focus on system property analysis. Furthermore, the goal is to identify and implement the necessary functionality that such a tool needs to support.**

An Enterprise Architecture tool that combines modeling and analysis is presented. Descriptions of organizations cannot only be created as a means of documentation but instead allow for reasoning over system properties.

1.2 Research contribution and delimitation

In this section, the research contribution made by the author is described in three steps.

First, the purpose of the performed research work is detailed, and measurable subtasks are stated.

Second, the contribution made by this research work is discussed.

Third, the author's specific contributions to the presented research work are discussed. This is necessary as the research work described in this thesis was carried out as part of several larger research projects. Each project had several contributors who sometimes also added to the presented research work. These contributions were coordinated and governed by the author of this thesis.

Purpose of the performed research

As stated in the previous section, the aim of the presented research project was to develop a software tool that can be considered a computer-aided design tool for

Enterprise Architecture. The goal was to develop a tool that uses models representing scenarios as input to analyze the characteristics of these setups and identify a preferable scenario. Fulfilling this goal included

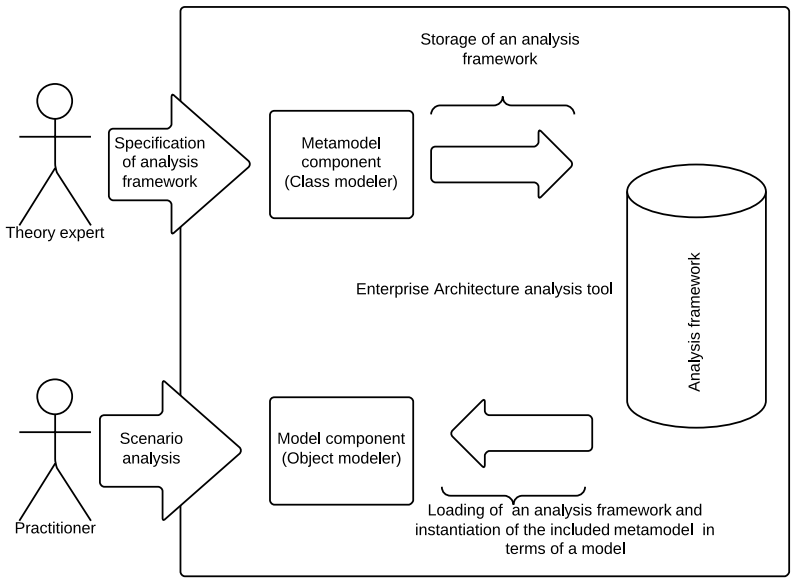
1. Eliciting the requirements of a tool for Enterprise Architecture analysis
2. Designing and developing a tool considering these requirements
3. Evaluating the quality of the created tool with regards to the identified requirements
4. Demonstrating this tool to both academics and practitioners

Contribution of the research work

Compared to existing tools, the proposed solution presented in this thesis allows in-depth analysis of the system properties of enterprise architecture models. The attributes of the modeled entities are calculated using mathematical reasoning. It is possible to trace chains of impact and find root causes to identify potential sources for improvement.

The simplified overall architecture of the tool is presented in Figure 1.1.

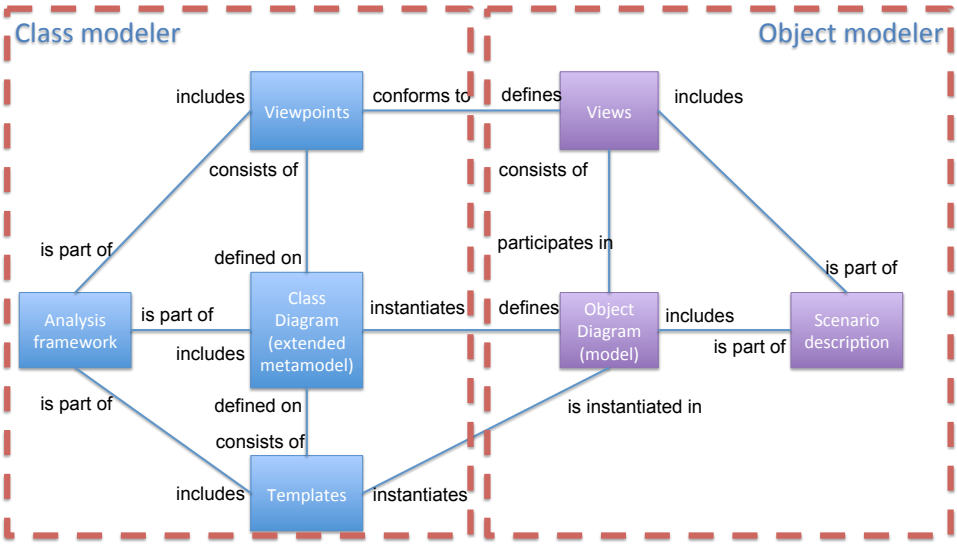
Figure 1.1: The Simplified overall architecture of the presented tool



It can be seen that the presented tool has two components, which are tailored to support two different target audiences. The first component, the class modeler, allows academics and other experts to iteratively develop analysis frameworks. These frameworks contain extended metamodels, specified as UML class diagrams [29], formalizing analysis theory. Additionally, frameworks may include templates, i.e., reusable blueprints that are specified on top of the UML class diagrams. Viewpoints, specifying coherent set of views based on the class diagrams, might be included in an analysis framework as well. These frameworks can describe various system properties, including cyber security, performance, modifiability and cost of usage. Using the second component, the object modeler, practitioners can apply these analysis frameworks to evaluations and facilitate decision-making. Thereby, the users instantiate the included, previously created UML class diagrams into UML object diagrams to create a model describing the scenario of interest. The user of the object modeler can instantiate the templates that are included in the analysis framework during the creation of the object diagram. Practitioners can also consider the resulting object diagrams based on views conforming to the viewpoints that are included in the analysis framework. Practitioners do not need to be experts in the fields that they strive to investigate but do need to know their IT and business domain.

The described concepts and the relationships between them are depicted in Figure 1.2.

Figure 1.2: The concepts included in the analysis frameworks and scenario descriptions



Enterprise Architecture tools typically do not address two different user groups.

Usually, these tools focus on the creation of company-specific descriptions, whereas no or only limited support is offered for the specification of analysis frameworks. Most of the available Enterprise Architecture tools can therefore be compared to the second component of the presented tool. The application of Enterprise Architecture is company-specific. Depending on the business sector that a particular company operates in, the goals of its Enterprise Architecture endeavor may vary. The banking sector often has a particular interest in availability questions, whereas power utilities frequently focus on cyber security aspects. Therefore, the tool allows analyses of various types.

Moreover, the tool is not limited to a set of properties that can be analyzed. Instead, the tool is implemented such that new types of analyses can be defined when needed. This can be performed by the tool user directly and does not need support from the tool developer. This is different than the functionally provided by many of the available Enterprise Architecture tools. These tools typically come with a fixed set of analysis capabilities. Additional analysis functionality cannot be added by the user; instead, he or she needs to purchase add-ons or enhanced versions of the tool to extend the analysis capabilities.

Using the presented tool, not only can analyses be added as needed but the existing analyses can also be modified and adapted to fit a particular company. This includes the setting of enterprise-wide default values, the adaption of the mathematics used during the analysis and the introduction of company-specific concepts that should be captured when creating Enterprise Architecture models. This is again different from many of the available Enterprise Architecture tools. If these tools possess analysis capabilities, these capabilities are often realized as a black box that cannot be modified by the tool user.

In addition, using the tool, it is possible to update the used analysis framework. In this way, even if they have been created in the past, models can be evaluated with regards to the latest theory. This tool separates the visual descriptions of an enterprise and the mechanisms used to evaluate these models. Taking this feature to the extreme, it is occasionally even possible to analyze a model with regards to a certain system property even if one did not have that particular aspect in mind when creating the model of the organization.

Another feature that typically cannot be found in other tools is the tool's ability to handle uncertainty. This is relevant with regards to two aspects. On one hand, practitioners might not know all of the details required to create a holistic model or might not be sure whether some aspects need to be described. The tool does not expect the models to be correct in every detail; instead, it is possible that a user expresses that he or she is unsure about the value of a certain attribute or relationship. On the other hand, the other user group, academics and theory experts, can also benefit from the consideration of uncertainty. In the present tool, this group can express uncertainty with regards to the analysis theory that they define. In this way, effects between attributes, default values or the existence of theoretical concepts can be specified, even if the experts are not completely convinced of the soundness of the theory they describe.

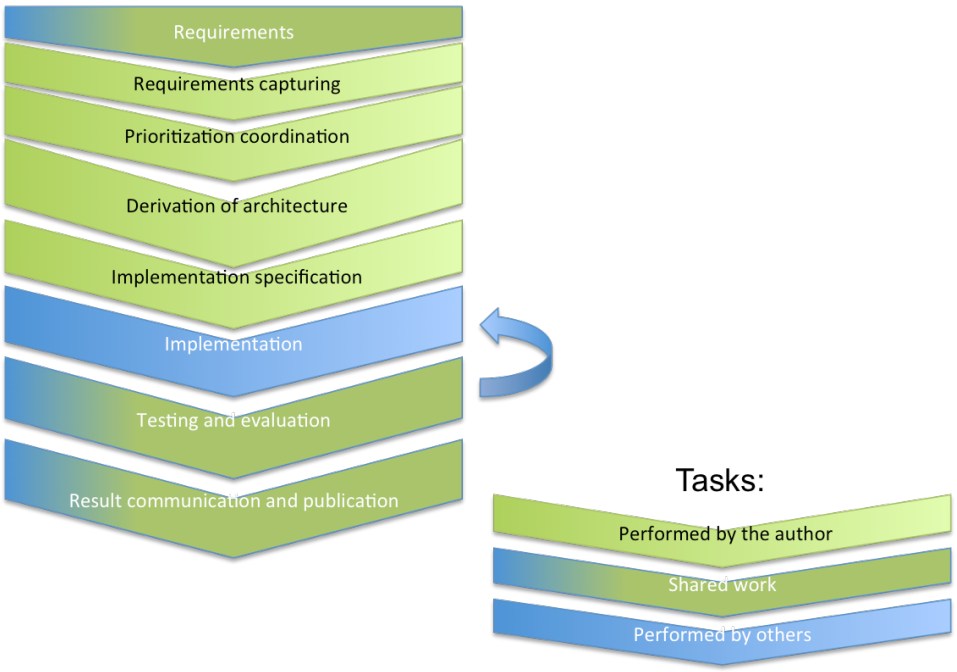
The author's specific contributions

During the performance of the research work described in this thesis, previous results were considered. The presented research work was part of several large research projects (cf. Section 1.3). Several authors and other contributors were involved in the presented project. In particular, the author did not implement the presented software tool by himself. This task was performed by a team of programmers. Instead, the author was responsible for the project management and worked in a variety of roles in numerous tasks. The specific contributions of the author are described in the following and are related to the work of others.

The first task of the described project was to design a development process that could be followed to reach the previously mentioned goal of the research work. To create an Enterprise Architecture analysis tool, relevant user groups were approached: academics interested in the specification of analysis frameworks and practitioners wanting to evaluate a certain architecture. Together with those groups, requirements for a tool for Enterprise Architecture analysis were derived. Thereafter, the collected requirements were translated into specifications that could be used during the implementation. The author was also responsible for the coordination of the requirements prioritization to select the next feature that should be added to the tool. Once a selection was made, the requirement was translated into a to-be (tool) architecture that could be used by the development team. This team consisted of one fulltime programmer and was frequently supported by students. The team did the actual coding on its own, coordinated by the author. The outcome was tested and evaluated by the author again. Additional implementation activities were triggered if needed. This was the case when tests failed or requirements were not met. The testing and especially the evaluation activities were performed together with the stakeholders of the requirements. Lastly, the outcome of the tool development was incrementally demonstrated together with the other stakeholders. This involved authoring scientific papers to present the tool features, presenting the tool to relevant audiences and conducting case studies together with industry partners.

The development process described above is visualized in Figure 1.3.

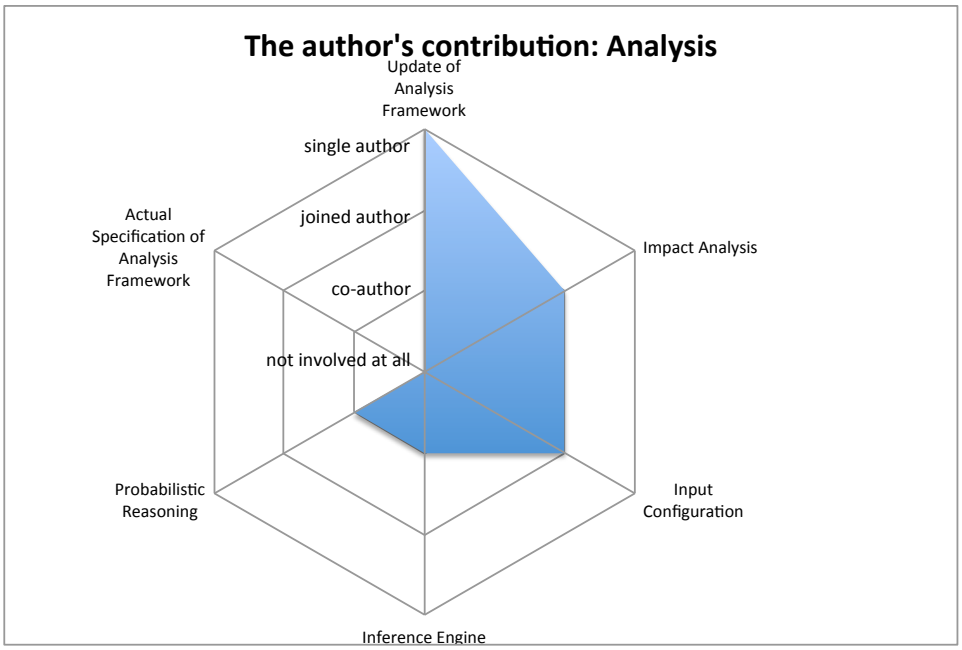
Figure 1.3: The role distribution during the development process



The author collaborated with numerous (academic) tool users to identify and realize relevant features. These collaborations were typically carried out in terms of larger research projects with several contributors. In particular, the research project on cyber security analysis, described in the following section (cf. Section 1.3), generated several requirements that were considered while performing the research work described in this thesis. In the remainder of this section, the author's contribution is discussed with regards to the two core aspects of the tool, analysis and modeling. Figure 1.4 and Figure 1.5 depict the contribution made. A scale with four categories, single author, author, co-author and not involved at all, is used to describe the contributions. The category "single author" refers to work on the tool in terms of feature development that the author carried out by himself, with very limited support from other contributors. The category "joined author" indicates that the author of this thesis was part of a group of equal contributors when a certain feature was developed. The category "co-author" describes features where the development was led by a project participant and the author of this thesis had a limited contributing role. Lastly, the category "not involved at all" describes features of the tool developed without any nameable contribution by the author.

The author's contribution with regards to the analysis capabilities of the presented tool

Figure 1.4: The author's contribution with regards to analysis



Six categories need to be considered with regards to the analysis of the capabilities of the tool. Chapter 7 contains a thorough description of these features (cf. Section 7.4). However, to discuss the author's research contribution, it is necessary to briefly introduce these categories here. The categories and the author's contributions are illustrated in Figure 1.4.

The category "update of the analysis framework" describes the capability of the tool to replace the framework that is used for the analysis with a reworked and improved version. The tool handles this exchange of the analysis framework internally; no manual reworking is required by the user. It is also not necessary for the user to restart his or her modeling endeavor. Instead, the user can continue extending the model originally created by applying a previous version of the analysis framework based on the latest version of that framework.

The category "impact analysis" covers the capability of the tool to identify dependencies between characteristics of the described system and how these characteristics impact the system properties that the user wants to evaluate. The tool is able to create networks illustrating the factors that impact the characteristics of a modeled entity. In addition, the tool can identify all aspects of a model that one

particular characteristic of a modeled entity impacts.

"Input configuration" describes the tool's capabilities to define the underlying theory to evaluate the characteristics of the created models. In this category, the tool's ability to express derivation rules based on and for the included model concepts are covered.

The inference engine is the component of the tool that evaluates derivation rules. These derivation rules are utilized to determine the characteristics of the modeled entities based on other characteristics that are part of the model. During the described research project, different engines using different algorithms to perform inference were designed and implemented (cf. Section 5.4).

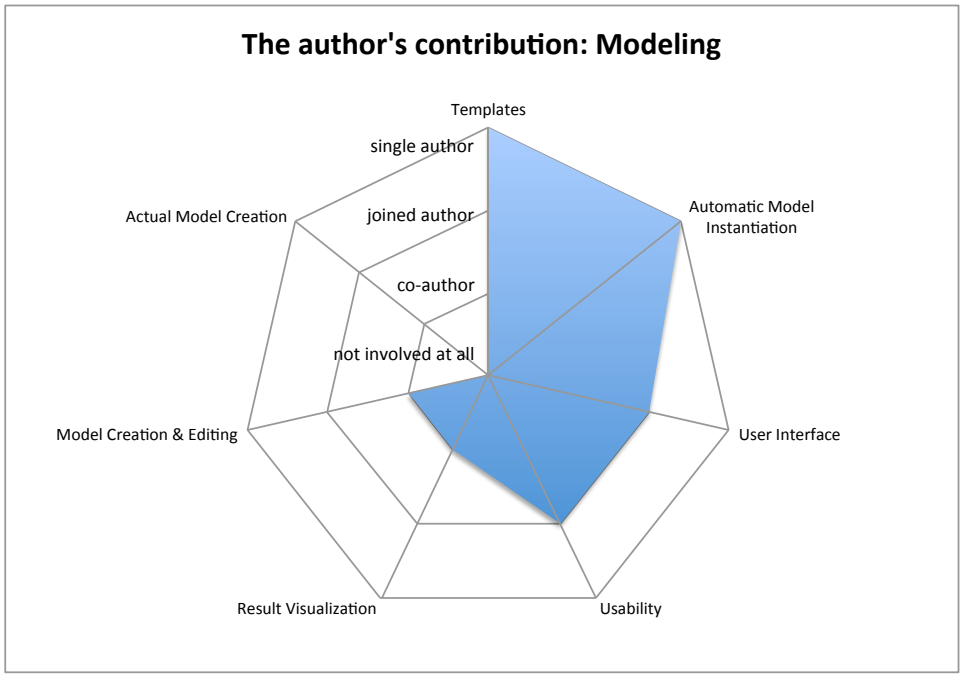
The category "probabilistic reasoning" describes the tool's ability to consider probability theory during the inference of the tool characteristics. Using probabilistic reasoning, the tool is able to feature structural definitional uncertainty, theoretical heterogeneity, causal uncertainty, empirical uncertainty and structural uncertainty (cf. Section 3.2). The category "actual specification of the analysis framework" describes the use of the tool to specify an analysis framework based on a previously established knowledge base. Before a framework can be specified, relevant concepts and their dependencies, in the context of the considered system property, need to be identified. However, the activity of deriving a knowledge base is outside the workflow supported by the tool. Instead, the tool is designed to create analysis frameworks representing such a knowledge base. Once an analysis framework is in place, it can then be used to evaluate scenarios of interest.

The author is the single author of the tool components allowing the updating of the analysis framework during run-time. Together with other project participants, he realized the need to conduct impact analysis and is therefore a joined author of this feature. He is also a joined author of the tool's feature allowing it to provide input to the analysis engine (input configuration), as this feature was added to the tool based on previous prototypical implementations of the tool implemented before the documented research project was initiated. The author is co-author of the inference engine, which was the result of a research project carried out at the author's department over several iterations. The author of this thesis is also co-author of the approach used to perform probabilistic reasoning that is used as part of the inference engine. This decision was made in research projects led by other project contributors. Lastly, the author did not directly contribute to the actual specification of analysis frameworks.

The author's contribution with regards to the modeling capabilities of the presented tool

For the analysis domain, seven dimensions need to be considered with regards to the tool's modeling capabilities. These dimensions and the author's contribution are illustrated in Figure 1.5.

Figure 1.5: The author's contribution with regards to modeling



The category "templates" describes the tool's ability to specify and use blueprints to accelerate the modeling process and reduce the visual complexity. Templates can already be defined as part of the analysis framework specification to guide the user during the utilization of the framework.

The category "automatic model instantiation" describes the import of data gained from external sources, the processing of these data in the preparation of an analysis and finally the creation of a model representing the processed external data.

The user interface describes the tool's graphical component used to interact with the user to create models. Interaction is bidirectional, as input from the user is processed and output in terms of visual depictions is created. This category also concerns dialogs, their structure and the look and feel used during the modeling endeavor.

The category "usability" covers the ability of the tool to provide the user with the information that he or she needs at a particular moment during the tool application. Additionally, this category addresses the tool's capability to support the fulfillment of the tasks that a user intends to complete. This category also captures the development of a workflow to be followed during the tool usage. Such a workflow suggests how to use the tool and in which order tasks should be executed.

The category "result visualization" describes the tool's abilities to graphically illustrate the results of a system property analysis.

Model creation and editing describes the tool features used to create models. Features to apply the analysis framework to describe a particular scenario are covered here as well. This category captures aspects such as the automatic generation of a layout of a model, the reuse of existing models to create a new one and the performance of changes in a model.

Lastly, the category "actual model creation" addresses the usage of the tool to actually investigate a scenario. This category describes the usage of analysis frameworks to describe a particular setup, evaluate this scenario and consider the analysis results.

The author was the single author of the components involving specifying and using the templates. He is also the single author of the tool's features allowing it to automatically create models (object diagrams) instantiating the class diagrams included in analysis frameworks. The author and other project participants are responsible for the design of the user interface and usability aspects; therefore, the category "joined author" is used. The author is co-author of the result visualization, which he designed, led by other colleagues. He also contributed to the development of the tool components that allow the creation and editing of the model under the leadership of other project participants. The author did not contribute to the actual creation of models to be used for analysis.

The discussed features of the tool are explained in-depth in Chapter 7.

1.3 Applications to cyber security

The tool presented in this thesis was partially developed as a contribution to a collaborative research project between the Swedish National Grid (Svenska Kraftnät), the Swedish Defense Research Agency (Totalförsvarets forskningsinstitut) and the Royal Institute of Technology. This consortium partially financed the research work described in this thesis. The goal of the joined research project was to provide a means for decision-support with regards to the design of industrial control systems from a cyber security perspective. The decision support should be provided so that information security is addressed from a holistic and enterprise-wide level. To fulfill this project goal, two areas had to be addressed. On the one hand, the complexity and size of industrial control systems (SCADA systems) had to be considered. On the other hand, the complexity of the cyber security domain needed to be addressed. Here, it is necessary to cope with various aspects, such as vulnerabilities, possible attacks and prevention techniques, to achieve a good level of security.

The presented software tool contributes to the fulfillment of this research goal for decision-support with regards to cyber security, as it offers an environment that can be used to perform system analysis. Some of the tool's features that are presented in this thesis support the analysis of enterprise models with regards to security in

particular. However, the tool as such is general and, as seen in the remainder of this thesis, supports the analysis of numerous system properties.

The connection between the described research work and the project on cyber security analysis can be traced throughout this thesis. Section 3.3 provides background information for this field of research. Furthermore, Section 5.6 describes the decisions made to specifically support the creation of enterprise-wide security modeling within the tool. In this section, several possible options to realize cyber security modeling are presented and compared. Based on this comparison, one modeling approach, attack graphs, was selected, and support for this approach was added to the tool. The usage of this support for the creation of models for cyber security analysis considering organizations from a holistic perspective is demonstrated in Section 8.2. The reliability of the design decision to make use of attack graphs is discussed in Section 10.2 as part of a larger discussion of the validity, reliability and generalizability of the performed research. Lastly, in Chapter 12, a section discussing future projects to further improve the tool's support for cyber security analysis is included.

1.4 Remaining structure of the thesis

In this section, the structure of the present thesis is briefly described. In Chapter 1, the topic of this thesis is introduced, the performed research is motivated and the main contributions made as well as the achieved results are discussed. The rest of this thesis unfolds as follows. In Chapter 2, the research methodology that was applied during this thesis is presented. Additionally, a mapping between the steps of the used research method and the chapters of this thesis is established. The underlying theory for the performed work is described in Chapter 3. In Chapter 4, the requirements of a tool supporting Enterprise Architecture analysis are discussed. Thereafter, in Chapter 5, different architectural options that were considered during the tool development and design decisions made are discussed. In the next chapter, Chapter 6, the development process that was followed is described. The outcome of this development activity is the topic of Chapter 7. Here, the tool is presented and visually illustrated. In Chapter 8, the aspects of how the tool was presented to a broader audience are covered. Practical applications at numerous companies as well as how the analysis frameworks were specified using the presented tool are described. The evaluation of the tool is described in Chapter 9. In particular, the fulfillment of the requirements presented in Chapter 4 is discussed. Thereafter, in Chapter 10, the performed research is discussed with regards to the validation, reliability and generalizability of the presented results. Following the used method, how the outcome of this thesis was communicated to relevant audiences is described in Chapter 11. Lastly, future work is outlined in Chapter 12. The thesis is concluded in Chapter 13.

Chapter 2

Research method

This chapter describes the underlying research design followed to achieve the goals of the presented research. In addition, this section describes how the structure of this thesis reflects the research design.

2.1 Design Science

The development of a tool for Enterprise Architecture analysis is a typical case of information systems (IS) research [241, 47, 271]. This discipline aims at developing IT artifacts, including constructs, models, methods and instantiations [114] . The Enterprise Architecture analysis tool that is described in this thesis can be classified as an instantiation, as the research resulted in the implementation of a prototype.

IS research is interdisciplinary, combining computer science, management, systems theory, sociology, finance, economics and anthropology [195].

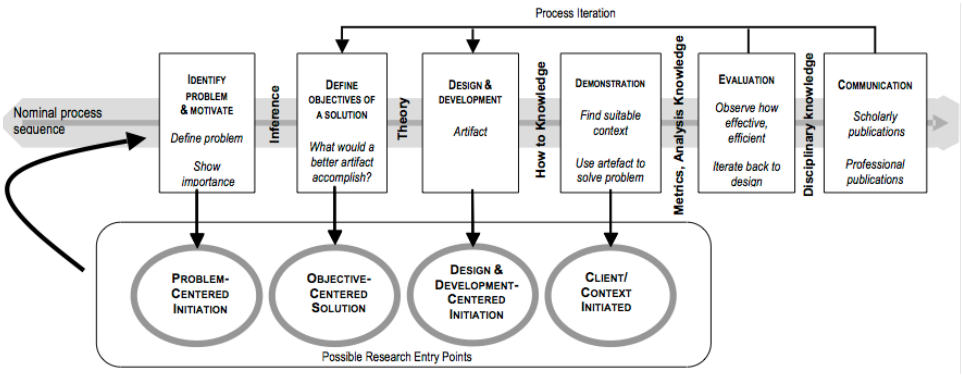
Gregor [96] identified five theories in the field of IS research, including the theory for design and action, also called Design Science [196]. Design Science addresses the question of how to do something [97]. In the context of the performed research, the question is how to create a tool for Enterprise Architecture analysis. It is about the principles of form and function, methods and justificatory theoretical knowledge that are used in the development of IS.

Design Science is a problem-solving paradigm originating from engineering [114]. It aims at the creation of artifacts based on existing kernel theories that are applied, tested, modified, and extended through the experience, creativity, intuition and problem-solving capabilities of the researcher.

To find the best, or at least a satisfactory, solution to the considered problem, Design Science is typically carried out as an iterative process. During each iteration, one tries to make use of the experience gained previously and create an even better solution [155, 114].

There are numerous processes outlining the performance of Design Science, including the following.

Figure 2.1: The Design Science Research Methodology (DSRM) Process Model [210]



In [211], the authors present, demonstrate and evaluate a Design Science research methodology (DSRM) process model that was derived based on prominent Design Science approaches. The process is visualized in Figure 2.1. This methodology was followed during the development of the Enterprise Architecture analysis tool presented in this thesis.

The methodology consists of six steps:

1. Identify problem & motivate
2. Define objectives of a solution
3. Design & develop
4. Demonstrate
5. Evaluation
6. Communication

A consideration of the steps in detail follows.

Step 1: Identify the Problem & Motivate

Initially, one needs to define the addressed problem. The user of the methodology will later use this definition to evaluate the created artifact. In line with [114] this, one must have a reason for conducting the research, i.e., there must be a need to propose a new artifact. This need can be expressed formally as the differences between a goal state and the current state of any type of system.

Step 2: Define Objectives of a Solution

The second step is to derive the objectives of a proposed solution considering the problem definition as well as what is possible and feasible. One should also consider the existing (not completely satisfying) solutions[114].

Step 3: Design & Develop

In this step of the methodology, one creates the artifact using the previously identified objectives as a theoretical basis. The authors of [211] stress that the resources required during the transition from objectives to design and development include knowledge of theory that can be brought to bear in a solution.

Step 4: Demonstrate

Here, one demonstrates the application of the previously developed artifact to solve one or more instances of the problem [261]. The usage can be demonstrated in several ways, including experiments, simulations, case studies and (mathematical) proofs [210].

Step 5: Evaluation

In this step, one observes and evaluates to what degree the artifact offers a solution to the identified problem [212, 195, 264]. Typically, one compares the objectives of a solution (step 2) to the actual results observed in step 4. Once again, the performance of this step should be tailored to the nature of the problem and the artifact. As part of this step, the method applicant can determine whether to iterate back to step three with the aim of creating an improved artifact or leave further refinement to follow-up projects. This decision might depend on the considered problem.

Step 6: Communication

The final step is to inform relevant audiences about the problem and its importance, the artifact, its utility and novelty, the rigor of its design, and its effectiveness [56]. This includes scientific publications. According to the authors of the DSRM "communication requires knowledge of the disciplinary culture".

2.2 The resulting Design Science methodology

This section describes how the previously described DSRM process model was used to achieve the goals of the research work presented in this thesis. For each of the method's steps, a mapping to the corresponding chapters of this thesis is established, and the content of these chapters is briefly presented.

Step 1: Research motivation for a tool for Enterprise Architecture analysis

This first step corresponds to the "Identify the Problem & Motivate" step of the DSRM process model. The first step of the DSRM is covered in the introductory

chapter of this thesis: Section 1.1 contains a description of the purpose of the performed research work i.e., the development of a tool for Enterprise Architecture analysis. The need for such a tool is motivated, and how it should differ from existing solutions is explained. Additionally, the context of the performed research is explained (cf. Section 1.3).

Step 2: Requirements on a tool for Enterprise Architecture analysis

The second step of the applied method reflects the step "Define the Objectives of a Solution" in the DSRM model. Combined, Chapters 3 and 4 cover this second step. Chapter 3 describes the existing and not fully satisfactory solutions and places these solutions in context. To do so, first, already available Enterprise Architecture tools are put into context. This helps reveal what is possible and feasible. In Chapter 4, the elicited requirements of a tool for Enterprise Architecture analysis are discussed. These requirements form the valuation criteria for evaluating the created artifact (cf. step 5).

Step 3: Development of a tool for Enterprise Architecture analysis

This third step is the instantiation of the "Design & Develop" step of the DSRM process model. Chapters 2, 5 and 6 cover this step. Combined, these chapters describe the design and development of the tool for Enterprise Architecture analysis. Chapter 2 (this chapter) contains a description of the method followed during the described research work. Chapter 5 describes the design of the artifact and explains the architectural options chosen and fundamental decisions made. This chapter also presents background information concerning the decisions made. Gathering this background information was relevant to understanding the potential architectural options and provided decision support for selecting the proper alternative. Chapter 6 contains a description of the development process followed to create the Enterprise Architecture analysis tool.

Step 4: Demonstration of a tool for Enterprise Architecture analysis

This step corresponds to the "Demonstrate" step of the DSRM process model. Chapter 7 describes the outcome, the artifact that was developed and especially the realization of the architectural options. Moreover, a description of a number of case studies using the Enterprise Architecture analysis tool can be found in Chapter 8.

Step 5: Evaluation of a tool for Enterprise Architecture analysis

In Chapters 9 and 10, an evaluation of the tool is presented, corresponding to the fifth step of the DSRM process model. In Chapter 9, this is first conducted by comparing the tool demonstrated in Chapters 7 and 8 to the objectives as stated in Chapter 4. In addition, the validity, reliability and generalizability of the performed research are discussed in Chapter 10.

Step 6: Discussion and communication of the achieved results

The final step of the DSRM ("Communication") is covered in Chapter 11, which describes how relevant audiences were informed and how the achievements made were presented.

The following chapter (Chapter 12) goes beyond the DSRM, discussing future work. This discussion is based on the results from the fifth step (Evaluation) and with regards to the feedback received while presenting the tool to relevant audiences. Lastly, Chapter 13 concludes the thesis and summarizes the performed work and achieved results.

Chapter 3

Enterprise Architecture and system property analysis

In terms of the research methodology presented in Chapter 2, this section contributes to the second step, "Define the Objectives of a Solution". In detail, an introduction to the topic of Enterprise Architecture is given. Thereafter, a method for Enterprise Architecture analysis is introduced. This section is followed by a section describing the currently available Enterprise Architecture tools and a consideration of their analysis capabilities. Lastly, this chapter contains a section on the properties typically considered during Enterprise Architecture analysis. These properties recur in the remainder of this thesis.

3.1 Enterprise Architecture

One widely accepted approach to IT management is Enterprise Architecture (EA). Many authors consider John Zachman, who developed the Zachman Framework [276] in the late 1980s, to be the founder of this discipline. He was the first to present a classification schema arguing for the structured description of enterprises in a holistic manor. In particular, the framework stressed the importance of modeling comprehensively from the strategic aspects of an organization via business and (IT) system details down to the implementational characteristics. Additionally, Zachman stressed the management aspects by suggesting six categories of models:

- The data description — What
- The function description — How
- The Network description — Where
- The people description — Who
- The time description — When
- The motivation description — Why

However, Zachman was not specific with regards to modeling notation. He suggested neither modeling syntax and semantics nor how to synchronize the different models.

Other Enterprise Architecture frameworks that were developed later, including TOGAF [111], feature metamodels, i.e., language specifications clarifying what to model and how do to so. TOGAF 9 contains the architecture development method (ADM) [109], which describes eight domains to be considered during an Enterprise Architecture endeavor. These domains cover, among others, the architecture vision, the resulting business architecture, the technical architecture as well as the projects performed to fulfill the architecture vision. With the release of the ArchiMate 2.0 [108] standard by the Open Group, a modeling language became available that "complements TOGAF in that it provides a vendor-independent set of concepts, including a graphical representation, that helps to create a consistent, integrated model ... which can be depicted in the form of TOGAF views [104]". The ArchiMate standard describes a mapping between TOGAF views and the ArchiMate viewpoints, going as far as that the "ArchiMate standard does not provide its own set of defined terms, but rather follows those provided by the TOGAF standard".

TOGAF and the Zachman Framework are some of the most commonly used Enterprise Architecture frameworks [139]. Other frequently used approaches include the Department of Defense Architecture Framework (DODAF) [191] for the United States Department of Defense, the Treasury Enterprise Architecture Framework (TEAF) published by the US Department of the Treasury [194] and the Federal Enterprise Architecture (FEA) developed by the Federal Government of the United States [54]. Including Enterprise Architecture frameworks developed by smaller organizations and those focusing on specific audiences, more than 900 frameworks and modeling tools provide different approaches to giving recommendations on what to consider while performing Enterprise Architecture endeavors [193]. Most of these frameworks have in common that they typically advocate for the description of companies with regards to the business, application, infrastructure and information domains. These four domains can be found under different names, as no common language for Enterprise Architecture has been established to date, and many Enterprise Architecture frameworks use their own wording to describe the offered concepts [231]. Furthermore, some frameworks aggregate some of the domains. For example, a consideration of the application and infrastructure is fairly common. In the business domain, it is common to describe organizations with regards to business processes, business roles, actors, business objects and products [108]. In the application domain, one can find such concepts as applications and data objects[108]. The infrastructure domain covers servers and desktop computers, network equipment and other physical devices[108]. How data between the three abovementioned domains is exchanged is covered in the information domain. This crosscutting domain both describes digital information stored in databases and processed in computer systems as well as the physical representation of this information, for example, as paper copies of contracts.

Many frameworks also emphasize the importance of modeling the visions and strategic goals of the described organization [66]. Keeping in mind that Enterprise Architecture is a tool for IT management with a special focus on business IT alignment, this is useful for evaluating whether the business domain and IT help reach these goals.

It is also common to find support for describing engineering requirements and, in particular, the expression of the requirements posed on different elements that are part of the organization. Often, requirements can be described for both business elements, e.g., that a certain business process should have a certain maximum execution duration and a certain infrastructure and that a certain server should be available during business hours.

In some frameworks, one can also find support for describing change projects that address the migration from one state of the enterprise to another. Often, one can describe these projects in terms of the team structure, resource allocation and the elements of the Enterprise Architecture that are affected.

Recently, the modeling of security aspects on an enterprise level has also become popular in some of the frameworks [213]. This is an aspect of relevance for both the business and IT domains of an organization. On the one hand, hacking and other forms of cyber criminality impact computer systems and the information those systems process. On the other hand, social aspects, including phishing and skimming, are relevant in the business domain.

Frameworks that are specific to the modeling process typically use metamodels to define the visual representations. These metamodels establish a language for the creation of models in terms of syntax and semantics. One finds the allowed concepts that can be used for the creation of enterprise-wide descriptions as well as how these concepts should be connected. Being specific about the language has the advantage that everyone who is familiar with a certain metamodel will be able to understand all models that instantiate it.

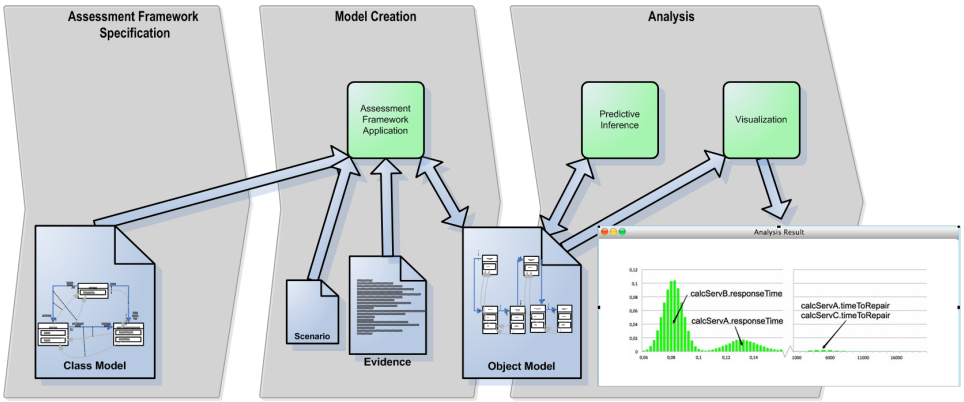
The various previously mentioned architecture aspects cover a variety of different topics from strategic business aspects to very technical details. To integrate this diversity, many Enterprise Architecture frameworks specify viewpoints on top of the metamodel [153]. Viewpoints typically define subsets of the metamodel in terms of allowed concepts and relationships. The result of the application of a viewpoint to a model is a submodel only capturing an excerpt of the whole. The subset is called a view.

3.2 Enterprise Architecture analysis

Kurpjuweit and R. Winter [148] identified that Enterprise Architecture models can be used for (i) documentation, (ii) design and (iii) analysis. In [133] the authors stress that Enterprise Architecture documentation and analysis posses no intrinsic value in themselves but can be valuable for providing good decision support to decision-makers concerning organizational-wide topics. However, only a few ex-

ceptions of the previously described Enterprise Architecture frameworks support analysis [137]. In [153] and [122] the authors use ArchiMate to conduct a performance analysis. [137] describes a method for Enterprise Architecture analysis. A further developed version of this method can be found in [43]. Despite minor differences, the underlying approach remains the same. The general thinking is that Enterprise Architecture models should be created with a purpose. It is too costly to collect all of the data needed for a model that is only used for descriptive purposes. Instead, modeling and data collection should be performed in alignment with the goals of the enterprise. The authors argue that potential scenarios should be evaluated with regards to their contribution to the goal of the organization. Doing so allows the comparison of scenarios, the identification of the most suitable one and decisions regarding its implementation. The authors of both previously mentioned Enterprise Architecture analysis methods argue for the usage of extended metamodels to create analysis frameworks. These extended metamodels not only describe the resulting models but also feature built-in analysis theory that can be used to evaluate system properties. The evaluated system properties should in turn be relevant for the goals of the enterprise. Figure 3.1 depicts the described method

Figure 3.1: The concidered Enterprise Architecture analysis method



The method in brief: First, an academic or other theory expert identifies a system property relevant for Enterprise Architecture analysis and creates an analysis framework consisting of an extended metamodel. This metamodel describes not only the allowed content and structure of the models that make use of it but also how the attributes of the model impact each other with regard to the chosen property to be evaluated. Following the method, the impact of the attributes on each other is expressed in terms of calculation rules. The creator of the extended metamodel specifies a set of parental attributes for each attribute as well as how the derivation of the attribute's value takes place based on the values of the parental attributes. Additionally, the relationships between the modeled entities and the

structure of the model can also be considered to derive the value of the modeled attributes.

For the attributes that are part of the extended metamodel, general data that describe them can be included as well. Typically, this is expressed in terms of default values. In the next step, practitioners identify scenarios of interest. These scenarios should be options with the potential of fulfilling the company's goals. Following the method, it is necessary to describe each scenario as a model instantiating the metamodel included in the analysis framework. This means that the practitioner should select a fitting analysis framework to evaluate a certain goal. For a particular scenario, general data can be replaced with specific information for some or all attributes that are part of the model. This results in a more specific description of the scenario. In the nomenclature of probabilistic inference, such instance-specific data are called evidence. In the following step, analysis, the practitioner performs inference to derive the quantitative values of the models' attributes and predict the characteristics of the modeled scenario. The authors of the method suggest inferring values using probabilistic reasoning. This allows the consideration of uncertainty. In particular, the authors of [138] have identified five areas of uncertainty: definitional uncertainty, theoretical heterogeneity, causal uncertainty, empirical uncertainty and structural uncertainty.

Next, the analysis results can be visualized. Additionally, beyond the scope of the proposed method lies the task of actually deciding which scenario to implement.

An example based on a real problem is as follows [1] . Närman et al. developed and published an analysis framework for availability analysis on an enterprise level [182]. The authors of [136] further developed it into a framework capable of evaluating such attributes as cost, data accuracy and modifiability. This framework is called the multi-attribute property (MAP) class diagram. A large Swedish bank found itself in a situation in which it was restricted by a tight budget but required its services to have the best possible uptime. The goals of the bank were thus high availability and low costs. A decision-maker decided to investigate one particular business service that was supported by a large network of IT services and other components. In particular, many load-balancer and redundant systems were present to ensure the availability of the business service. The decision-maker used the MAP metamodel to describe the business service and IT support. He developed two scenarios, represented in two models. Both models instantiated the extended metamodel included in the already mentioned analysis framework, containing analysis theory for, among others, cost and service availability. One model covered the as-is situation, including the mentioned redundancy and load-balancing techniques. The second scenario and model described a flattened architecture without redundancy. For both models, the decision maker collected evidence to describe the costs as well the availability of the systems. Next, he used the tool for Enterprise Architecture analysis that is the result of the research work presented in this thesis

[1] A course participant presented a slightly modified variant of this example as part of an enterprise architecture course supervised by the author.

to evaluate both models. The tool calculated the values for the availability and cost of the business service and visualized the results. The decision-maker realized that he could achieve almost the same degree of availability in the second scenario, but at a markedly lower price.

Enterprise Architecture tools

Enterprise Architecture models often become very large. They can contain several thousands of entities and an even larger number of relationships between them [15]. Therefore Enterprise Architecture tools are used to handle this complexity. Some of the most well known Enterprise Architecture tools include Rational System Architect [124], ARIS Architect & Designer [2], BiZZdesign Architect [27], the Troux Architect [256] and planningIT [6]. Gartner [88] and the Department of Software Engineering for Business Information Systems at Technische Universität München (TUM) [166] identified more than 10 tools that organizations often use, but the market is even larger. In their latest survey, TUM considered more than 50 tools [277]. These tools generally support one or several of the previously mentioned frameworks and their metamodels, whereas some have their own framework and metamodels.

Analysis capabilities of common Enterprise Architecture tools

According to the Enterprise Architecture tool studies performed by Gartner and TUM, the previously mentioned tools possess some analysis capabilities. These analyses are typically qualitative and based on visualizations and do not include the consideration of system properties. For example, which IT-systems support a selected business process, how many applications read a certain data object or which roles are assigned to a specific department can be investigated. In addition, two versions of a model (typically as-is and to-be) can be visually compared.

Within these tools, Enterprise Architecture models usually cannot be analyzed with respect to such properties as performance, business fit, availability or cyber security. Common Enterprise Architecture tools allow the relating of entities, indicating that a certain concept, e.g., is a specialization of another one, is composed of several concepts or is the predecessor of a modeled object. Within the tools, relationships visually indicate that the real concepts, which the model entities are meant to reflect, have a connection. A certain organizational role might be a specialization of another one, a business process might consist of several sub-processes or an activity might be the predecessor of another one. However, within the tools, the relationships are merely illustrative and are not used to analyze the causal impact of the properties of the modeled entities on each other. The tools do not infer the state of a considered attribute based on other attributes that are affecting it. Third, as well-known Enterprise Architecture tools do not focus on advanced analysis capabilities, the tools do not cover the aspect of incomplete models. These tools only consider elements that have explicitly been modeled. The fact that the

tool user might not know all of the details for creating a holistic model or might not be certain of whether some aspects need to be described is not covered. Lastly, well-known tools expect the models to be correct in every detail. A user typically cannot express that he or she is unsure about the value of a certain attribute or relationship.

In contrast, Abacus [62] offers complex analysis capabilities. It allows the investigation of Enterprise Architecture models with respect to performance, reliability, cost, openness, complexity, alignment regulatory and modularity [12, 13]. This is performed using discrete-event and Monte Carlo simulations. Moreover, tradeoff and sensitivity analysis can be performed. Abacus lacks two characteristics of the tool presented in this thesis [2] . First, Abacus does not allow the incorporation of definitional uncertainty, theoretical heterogeneity, causal uncertainty, empirical uncertainty and structural uncertainty in the analysis. Second, Abacus has a fixed set of system properties that can be evaluated; the tool presented in this thesis allows the user to specify new or modify current analysis theories.

AnnL [156] is another Enterprise Architecture tool possessing advanced analysis capabilities. It has a focus on the planning of viable enterprises; in particular consequence analyses can be conducted during the design process of enterprises. AnnL helps to measure the risks of the options that have been generated from combinations of alternative resources. It also allows the determination of the price risk of the options through a cost model and the description of an action plan for carrying out the tasks using these resources. AnnL evaluates a system dynamics model using Powersim Studio 9 [245] based on input specified in Excel spreadsheets [169]. The result of the analysis is then again presented in Excel. Compared to the tool presented in this thesis, AnnL lacks several characteristics. For example, AnnL does not support graphical modeling. The used metamodel is fixed and cannot be adapted by the user. Moreover, AnnL focuses on the risks that arise during the design. The actual identification (based on, for instance, a cyber security analysis) of risks is not supported. Lastly, AnnL allows the modeling of the impact of attributes on one another and the extension of this impact. Uncertainty about whether this impact relationship exists at all cannot be expressed.

3.3 Properties for Enterprise Architecture analysis

In this section, relevant properties that are often considered for Enterprise Architecture analysis are introduced. In accordance with the applied research methodology (cf. Chapter 2), this section contributes to the problem definition. A tool for Enterprise Architecture analysis should support the types of analyses that potential tool users likely want to perform. Therefore, it is important to understand these

[2]The presented weaknesses were identified based on the publicly available information about Abacus on the vendor's webpage and other secondary sources. The author tried to register for a trial license to perform an in-depth evaluation of the tool. The request was denied by Avolution, the tool vendor, due to a conflict of interest between their tool development and the tool development performed during the described research work.

potential users. Moreover, this section serves as preparation for Chapter 8, in which applications of the tool are reported. These tool applications were all conducted with the purpose of either specifying an analysis framework to analyze one of the following properties or applying such an analysis framework to evaluate a system property.

Cyber security

The usage of off-the-shelf products, interconnected information systems and increased complexity have given rise to the emergence of logical vulnerabilities over the last several decades [164, 34]. The impact of several recent large-scale cyber-attacks and their capability made this problem obvious [151].

For an organization-wide decision-maker, this is not just a question of protecting soft- and hardware systems. Social aspects, including phishing, scamming and baiting [67, 173], turn cyber security into an issue that is equally important for the business level and, in particular, for the employees of a given company. One other important aspect is that cyber security attacks are typically conducted in terms of parallel or sequentially conducted attack-steps aiming to reach a certain goal. In many cases, an attacker does not stop after having performed one single step but rather uses one step after another to gain more privileges and fulfill his target.

When Enterprise Architecture models are used for cyber security analysis, they help evaluate the company as a whole and identify likely sequences of attacks as well as weak links. A systematic consideration of these weak links can be useful for improving the level of cyber security.

Application modifiability

One of the core concerns of Enterprise Architecture is to ensure that business environments and IT are in alignment. In particular, this means ensuring that the IT domain is able to respond to changes triggered from the usually volatile business domain. If the business changes, for example, when a company launches a new product or modifies a business process based on customers' feedback, the IT systems need to be modified to continue to support the affected processes. Possible changes include extending, phasing out, adapting and restructuring the enterprise systems [18].

At one extreme are local modifications that only impact one particular system. On the other extreme is the redesign of the entire system landscape by, for example, introducing a service-oriented architecture. Many information systems are currently interconnected, and it is necessary to use them jointly to achieve a certain result. This connection also needs to be considered when performing modifications. A change in a system might cause a chain reaction requiring further changes at additional systems. These changes in turn might trigger follow-up changes at even more systems, and so on.

Performing changes ad hoc might be a good approach for solving abrupt problems in some cases; however, if those changes are not followed up, they are likely to negatively impact future modifications. According to [136], these raise questions for decision makers, such as the following: "Is the source code easy to grasp?" "Which systems are interconnected and how?" "Are the systems too complex?"

Using Enterprise Architecture models for modifiability analysis supports answering these questions helps decision-makers determine how much effort a given modification to an enterprise information system would require.

Data accuracy

Not only digital information but also physical documents and other artifacts capturing information are transferred within an organization and to connected enterprises. Typically, applications or persons read, write, update or delete objects (physical or virtual) that contain data of relevance. The processing of digital information, which originates from analog sources, including paper documents or fax messages, might be the most illustrative example of the close connection between the business and IT domains.

The use of low-quality data in IT systems is costly according to [254]. Analyzing the quality of data is therefore of great importance for reducing waste. Completeness, consistency, currency, relevance and accuracy are the most common dimensions of data quality [216].

For Enterprise Architecture analysis, it is common to focus on data accuracy [136]. IEEE defines data accuracy as (1) a qualitative assessment of correctness, or freedom from error, and (2) a quantitative measure of the magnitude of error [215]. Accurate data is the basis for sound decisions. It does not matter whether the data is used in a manual business process or within an automated application service; both applications are executed flawlessly at all times, resulting in steadily deteriorating data accuracy. Another widespread phenomenon is the fact that the further away from the source the data are, the poorer their accuracy will be [136].

Supported by Enterprise Architecture analysis a decision-maker can evaluate the accuracy of the exchanged facts, concepts or instructions in a holistic manner. It is possible to identify humans or IT systems operating on a certain data set and to evaluate whether they contribute to an improvement or decrease in the data accuracy.

Application usage

In medium and large enterprises, the usage of a large application portfolio is common. Often, these portfolios represent large investments and consist of several hundreds of applications. Realizing their full value is difficult in many cases [35].

Many organizations are facing the problem of an uncontrolled proliferation of applications. This leads to a heterogeneous application portfolio, redundancy, expensive IT systems and, in particular, poor business-IT alignment [224].

Enterprise Architecture provides means for company-wide, structured application portfolio and landscape management [221].

Service availability

The availability of business processes or business services and thus continuous business operations depends on the availability of the application and infrastructure services they consume [236].

In [79], the authors identified the relevance of this aspect for decisions. Unavailable IT systems not only incur direct costs [237] but also intercept business operations and have a negative impact on the market value of publicly traded companies [23].

Interoperability

According to IEEE, interoperability describes the capability of two or more systems or components to communicate with one another, typically by exchanging information [215]. For a decision-maker, ensuring interoperability translates into the task of satisfying a set of communication needs. Enterprise Architecture analysis provides a means to identify and fulfill these needs throughout the organization.

Cost

Gartner reported in 2007 that information system costs consumed 4.4 % of European firms' revenue [86]. They further concluded that the information system costs were likely to increase. The Bureau of Economic Analysis in the United States reported that the share of IT in business equipment investments exceeded 50 percent in the year 2000 [192].

Considering the huge amount of money spent and the importance of information systems (cf. Chapter 1), it is imperative to increase the quality of decisions concerning information system management.

Bad decisions not only negatively impact the smooth operation of a company but also have high costs. Enterprises typically procure IT systems with the intention of using them over a long time period [4]. Thus, considering the longevity of information systems, it is not sufficient to only consider the initial costs. Instead, one should also consider maintenance costs, support costs, the cost of changes and license costs as well as the annual costs. Looking at an organization as a whole, other cost drivers outside the IT domain are worth monitoring, including labor costs, the cost of used products and the costs that arise from the consumption of (business) services that are externally provided.

Using Enterprise Architecture analysis for cost analysis helps identify spending, allows the price of a product to be traced to the IT systems used during its manufacturing and can be used to evaluate the application portfolio of an organization [10].

The authors of [272] state that "an important application of cost analysis techniques is the calculation of IT-related costs and the allocation of these costs to products, services, processes, organizational units and other artifacts on the strategy layer and on the organization layer".

Multi-property utility

Multi-property utility evaluation aims at determining how satisfying a given solution is [19]. This system property is of particular interest when a decision-maker tries to evaluate multiple other properties, as discussed in this section, at the same time. In particular, utility theory can be used to perform trade-off analysis between otherwise incomparable properties, such as cyber-security and interoperability, to identify the best outcome [142]. Typically, a utility function is used to evaluate a particular scenario. This utility function takes all of the properties of interest into consideration and evaluates them, yielding one final score per scenario. The literature typically describes this phenomenon as multi-attribute utility theory [165, 126].

Enterprise Architecture analysis can incorporate many properties. Often, the decision-maker has to balance these properties against one another when trying to find the best possible architecture. Multi-property utility theory applied to Enterprise Architecture analysis helps the user optimize this tradeoff.

Actor profitability

Business actors, such as companies, follow one or several business models, depending on their size and the products they deliver. Frequently, they are part of business collaborations with other organizations. When considering both business models and collaborations, it is important to predict the performance of the business-to-be before actually implementing it.

Using such techniques as e-3 value modeling [95], it is possible to investigate such properties as costs, revenues, risks and profitability and evaluate the economic efficiency in advance. Additionally, potential business rules can be tested and sensitivity analysis conducted to identify strengths and weaknesses.

Business performance and organizational structure

To achieve an organization's goals, it is important that its structure is designed in the most supportive way. To evaluate the quality of an organizational structure, Mintzberg [174], based on numerous case studies, suggests the consideration of organizational learning, motivation, efficiency, productivity, coordination, flexibility and variability.

Analyzing the organizational structure using Enterprise Architecture models, an IT decision-maker can evaluate whether a proper mapping between organizational goal and structure exists. He or she can additionally use this type of analysis to derive a supporting business structure and underlying IT architecture.

Chapter 4

Requirements on a tool for Enterprise Architecture analysis

In this chapter, the "Define the Objectives of a Solution" step of the applied research method is continued. In particular, the requirements of a tool for Enterprise Architecture analysis are described. Three domains were considered when identifying these requirements: the criteria that the department of Software Engineering for Business Information Systems at Technische Universität München (TUM) [166] and Gartner [88] used in their Enterprise Architecture tool evaluations; the method for Enterprise Architecture analysis, which the tool presented in this thesis is meant to support; and the requirements of CAD (computer-aided design) tools and expert systems. Consideration of the requirements of Enterprise Architecture tools is evident, as the described research work aimed at advancing the field of Enterprise Architecture tools. Therefore, a new contribution in this field should be at least as good as the already available tools. To identify whether this is the case, the same evaluation applied to existing Enterprise Architecture software is used. The presented tool was developed with the goal of creating an Enterprise Architecture tool with a focus on analysis capabilities. Therefore, Enterprise Architecture analysis, in particular a method for doing so, is a relevant source of requirements as well. The aim of creating an Enterprise Architecture analysis tool arose in part while considering existing solutions outside the Enterprise Architecture domain that combine modeling and analysis. In particular, CAD tools were role models, successfully demonstrating that visual descriptions do not necessarily need to be descriptive but also can be used to evaluate the properties of the system they are capturing. This section ends by summarizing the elicited requirements. This summary will be used in Chapter 9 to evaluate the tool.

4.1 Requirements derived from Enterprise Architecture tool evaluations

As mentioned in Chapter 3, some groups have performed Enterprise Architecture tool evaluations. In this thesis, the work of TUM and Gartner is highlighted. These two groups have different perspectives on the field of Enterprise Architecture. TUM has an academic background, whereas Gartner has a stronger link to Enterprise Architecture users from industry. The work of these two groups was chosen for study to obtain a holistic perspective of the evaluation of Enterprise Architecture tool evaluations.

Both groups evaluate Enterprise Architecture tools according to eight dimensions. TUM's academic background is also indicated by the criteria they consider when investigating tools: importing, editing and validating; creating visualizations; interacting with, editing, and annotating visualizations; the flexibility of the information model; communication and collaboration support; support for large-scale data; impact analysis; and reporting and usability. Gartner, in contrast, evaluates tools with regards to repository/metamodel, modeling, decision analysis, presentation, administration, configurability, frameworks and standards and usability. A brief consideration of the categories, starting with TUM, follows.

Importing, editing, and validating

The category importing, editing, and validating describes the tool's ability to make use of data from external sources. This includes the validation of the imported data with regards to its data quality. The authors stress the importance of import capabilities, as many companies create Enterprise Architecture models using data from existing models (describing subparts of the enterprise) or other sources [70, 225]. Editing describes the ability to modify data during the import process to make it fit the rest of the model.

Creating visualizations

The creating visualizations category covers the presentation functionality of the Enterprise Architecture tools. Aspects such as the capability of depicting the selected relationships between entities, e.g., a part-of relationship using containment or a vertical/horizontal alignment of symbols, are considered as well. Additionally, user-defined visualizations, with or without templates, are investigated. For TUM, the term "template" refers to a type of visualization, not an instance; thus, defining a new template creates a new type of diagram with rules defining the entities to be displayed and how they shall be displayed.

Interacting with, editing of, and annotating visualizations

Interacting with, editing and annotating visualizations covers the tool's capability to handle visualizations. Enterprise architecture has a strong focus on visualization and graphically creating an overview. Therefore, such aspects as zooming in and out of visualizations, layering or the closing of symbols to hide a complex inner structure are of particular relevance. Additionally, the ability to edit visualizations is considered. Annotating visualizations with additional information is also of relevance in this category. TUM also considers results from metric evaluations as annotations.

Flexibility of the information model

TUM uses the flexibility of the information model category to express that the underlying metamodel of the tools should remain adaptable, in particular after a (partial) instantiation. TUM considers the ability to create, edit and delete the entities, attributes and relationships of the predefined metamodel to create a company-specific modeling language. TUM also evaluates whether the tool users can perform the metamodel configuration or if this is a task exclusively performed by the tool vendors. Export of the metamodel and (re-)import are captured as well. In addition, TUM investigates whether attributes can be set to default values and whether information can be marked as mandatory.

Communication and collaboration support

The communication and collaboration support category describes the collaboration support of the tools. Whether multiple users can edit the models in parallel is considered, as are the available auditing mechanisms and workflow or notification capabilities. User access management and the detail in which this can be accomplished are also part of this category.

Support of large scale data

Support of large-scale data considers whether the tools can cope with models of several thousand entities and all of the connections between these entities. TUM investigates two particular aspects: the performance and scalability of the tool as well as navigation within huge data sets.

Impact analysis and reporting

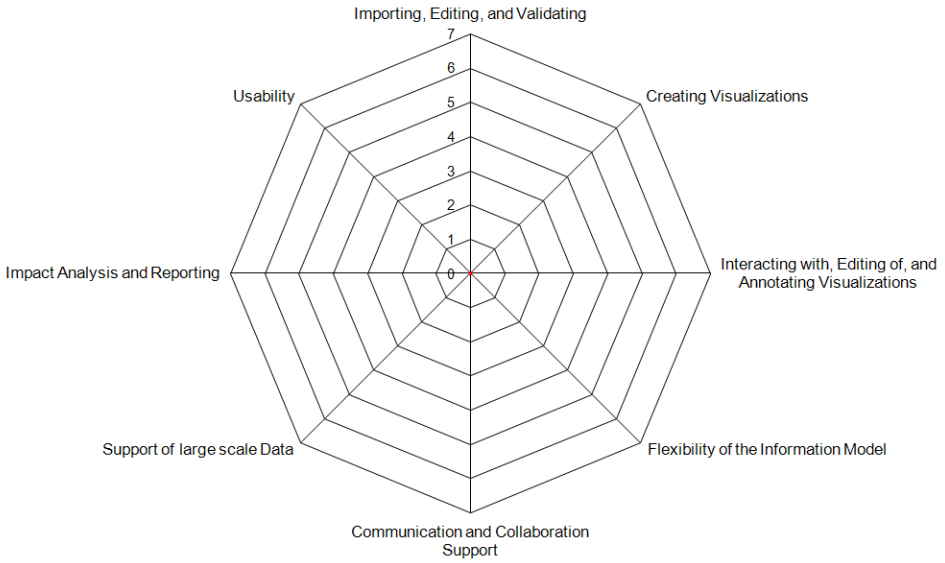
This category refers to the possibility of performing calculations or impact analysis based on the models. TUM stresses that calculations in this specific case refer, e.g., to summing up or deriving averages of values of an entity's attributes as well as of a transitively linked entity. Furthermore, impact analyses involve traversing specific relations and filtering the resulting data according to the given criteria. Another

aspect is whether these features are part of the user interface or accessible using a query or programming language. Additionally, whether the tool can produce reports and whether filters can be used during the report creation activity are considered.

Usability

The usability category describes the overall user experience. In particular, TUM investigates how simple or complicated it was to use the provided functionality. The considered aspects include intuitive and well-structured menus with self-explanatory names, consistent editors and supportive help systems. This dimension is based on a subjective impression of the tool usage.

Figure 4.1: The Kiviat diagrams used by [166]to evaluate Enterprise Architecture tools



TUM uses the described categories to draw Kiviat diagrams (cf. Figure 4.1) allowing comparison of the investigated Enterprise Architecture tools. Each axis reflects one of the previously discussed categories.

The following describes the categories that Gartner uses in their Enterprise Architecture tool evaluation [89].

Repository/metamodel

In Gartner's evaluation, the category repository/metamodel refers to capability of Enterprise Architecture tools to store models instantiating a metamodel and to make this model available for the tool users. Gartner stresses that Enterprise Architecture models should typically capture at least five aspects: the enterprise context, business architecture, information architecture, technology or technical architecture and solution architecture. Also of relevance are the possibility of modifying and customizing the metamodel, the use of frameworks on top of the selected metamodel and the possibility of differentiating between geographic and organization levels.

Modeling

Gartner's modeling category addresses the aspect of presenting information to different stakeholders. This includes the creation of a holistic model illustrating an enterprise from "the strategy level to technical implementation"[89]. It also covers the creation of models of the as-is state as well as one or more to-be states. Another aspect is the traceability of changes. Enterprise architecture tools should be able to indicate the impact of (business) changes to all architecture viewpoints. Interactions with other modeling tools are also considered in this category.

Decision analysis

In the decision analysis category, Gartner evaluates whether the Enterprise Architecture tools are capable of performing gap analysis between the current and future states of the architectures. Additionally, Gartner investigates whether what-if analysis for tracing the consequences of changes can be performed, allowing the evaluation of business, organizational and regulatory compliance, among others. Additionally, investment management support, project and portfolio management functionality and strategic panning capabilities are part of this category.

All of these aspects aim at fostering a scenario planning and system-thinking approach.

Presentation

This category addresses the visualization of the information captured by Enterprise Architecture tools and how this information is accessible to interested stakeholders. Enterprise architecture models might be large and comprehensive; however, Gartner stresses that the presentation of the contained information must be straightforward. This presentation should be illustrative and applicable to the target audience regardless of background. Areas of interest are visualizations that make it easy to identify the impacts of changes and support multiple scenarios depending on the stakeholders. Moreover, interfaces to other presentation applications are also relevant.

Administration

The administration category aims at evaluating whether Enterprise Architecture tools can strike the balance of supporting large-scale models while still having a clear and user-friendly graphical user interface. The aspect of providing accurate information is considered as well. Additionally, collaborative usage of the tools and user access management are considered.

Configurability

Gartner analyzes whether Enterprise Architecture tools can be adapted to specifically fit the organizations using them. This includes modifications of the metamodel, visualizations and the tools' default language. Furthermore, Gartner investigates the ease of performing a configuration and whether multiple environments (development, testing and production) are supported.

Frameworks and standards

Using this category, Gartner evaluates whether Enterprise Architecture tools support the Enterprise Architecture endeavor according to specific frameworks and standards, such as TOGAF, the Zachman Framework, FEAF, MODAF and DODAF. One area of interest is to evaluate whether Enterprise Architecture tools allow the use of several of the previously mentioned standards and frameworks in parallel. Gartner also investigates whether these standards and frameworks can be modified in accordance to the organizations' approach to working with EA.

Usability

Gartner investigates the tools' ability to address complex models in a simple manner. This category also captures whether the tools are easy to use, intuitive and straightforward to learn and maintain. This includes the user interface, the modeling domain and the administrative functionality typically happening in the background. Another aspect is how easy it is to remember how to use a tool.

Tool survey summary

Considering the eight criteria considered by TUM and the eight criteria considered by Gartner, one can notice that the two sets do not evaluate the same aspects. Both evaluations overlap; however, Gartner has a stronger focus on management aspects in terms of change propagation and usage of standards as well as visualization for specific stakeholders. In addition, the organizational-wide usage of the tools is considered by the evaluation of administrational aspects. TUM focuses more on modeling, model creation from existing data sources and support for large models.

Both evaluations investigate the usability of Enterprise Architecture tools, using even the same name for the category. Additionally both investigate the ability to

trace changes and create reports over impacted entities. TUM calls this category impact analysis and reporting whereas Gartner uses the term decision analysis. There are some categories do not match exactly, but are related to one another. Gartner's administration category, a tradeoff between support for large data and user-friendliness, covers among others large models, as described by TUM's Support of large-scale data. Moreover, Gartner's presentation category addresses the presentation of the right information for the right stakeholders. This aspect is considered spread over three categories by TUM (creating visualizations, interacting with, editing of, and annotating visualizations, communication and collaboration Support). TUM's category flexibility of the information model is partly addressed by Gartner's Frameworks and standards configurability describing the modification of the used metamodel, as well as Gartner's configurability addressing the adapting of the Enterprise Architecture tools to fit a company's purpose.

Furthermore, some categories can only be found in one evaluation. The category of importing, editing and validating describes the usage of data from existing sources and is only part of TUM's evaluation. Gartner's repository/metamodel category is not explicitly reflected by TUM. However, indirectly, TUM assumes a structure of metamodels describing the content of instantiating models as well as an underlying architecture storing this information. Gartner's category modeling is not explicitly part of TUM's evaluation, either. Nevertheless, TUM assumes a model to be in place covering everything from the organizational goals to the technical details implemented to reach that goal.

The following table (cf. Table 4.1) depicts the described comparison.

TUM	Relation	Gartner
Usability	Equal to	Usability
Impact Analysis and Reporting	Equal to	Decision analysis
Support of large-scale Data	Part of	Administration
Creating Visualizations Interacting with, Editing of, and Annotating Visualizations Communication and Collaboration Support	Part of	Presentation
Flexibility of the Information Model	Spread over	Frameworks and Standards Configurability
Importing, Editing, and Validating	Unique	
	Unique	Repository/metamodel
	Unique	Modeling

Table 4.1: Enterprise architecture tool survey comparison

Table 4.2 summarizes the requirements deduced after comparing the two discussed Enterprise Architecture tool surveys. The categories included in Table 4.2, entitled EA tool survey comparison, were rephrased to foster an evaluation of the presented tool against these categories (cf. Chapter 9). Additionally the criterion "the tool shall support an extendable metamodel" was introduced to replace the "flexibility of the information model" category from TUM, which corresponds to two categories identified by Gartner, "frameworks and standards" and "configurability". Moreover, Gartner's combined criterion "repository/metamodel" was divided into two requirements to enable an evaluation of this criterion. The two requirements "the tool shall support the storage of models instantiating a metamodel in a repository" and "the tool shall support the creation of metamodels covering the domains of business architecture, information architecture, technology or technical architecture and solution architecture" were introduced to replace "repository/metamodel".

Requirements derived from Enterprise Architecture tool evaluations
The tool shall offer a high degree of usability
The tool shall possess analysis capabilities
The tool shall possess administrative capabilities
The tool shall possess presentation capabilities
The tool shall support an extendable metamodel
The tool shall support the import, editing and validation of data from external sources
The tool shall support the storage of models, instantiating a metamodel, in a repository
The tool shall support the creation of metamodels covering the domains business architecture, information architecture, technology or technical architecture and solution architecture
The tool shall support the creation of models

Table 4.2: Requirements derived from the considered Enterprise Architecture tool surveys

4.2 Requirements derived from an Enterprise Architecture analysis method

Section 3.2 describes a method for Enterprise Architecture analysis. The aim of the research work presented in this thesis is to develop a tool that supports this method. To make it possible to evaluate whether the presented tool supports the method, the included method steps need to be considered. In particular, it is necessary to investigate whether the resulting tool supports all of the method steps properly. To foster an evaluation of the proper support of the steps, they need to be translated into requirements. Following the method for Enterprise Architecture analysis, first, an academic or other experts identifies a system property relevant for Enterprise Architecture analysis and creates an analysis framework including an extended metamodel. Thus, the requirement "the tool shall support the creation of extended metamodels describing system properties" was identified.

According to the method, this metamodel describes not only the allowed content and structure of the models that make use of it but also how the attributes of the model impact one another with regard to a chosen property considered for architecture analysis.

The impact of attributes on one another should be expressed in terms of calculation rules. This aspect of the model led to the requirement "the tool shall provide means to express the impact of attributes in terms of calculation rules".

Following the method, the creator of the extended metamodel specifies a set of parental attributes for each attribute as well as how the derivation of the attribute's value takes place based on the values of the parental attributes. In addition, the

method suggests that relationships between the modeled entities and the structure of the model should be considered to derive the value of the modeled attributes. To cover this aspect, the requirement "the tool shall consider parents of an attribute and the structure of the model during the derivation of attribute values" was elicited.

For the attributes that are part of the extended metamodel, the considered method for Enterprise Architecture analysis suggests that general data that describe these attributes can be included as well. These data should typically be expressed in terms of default values. To cover this aspect, the requirement "the tool shall support the specification of default values for attributes" was elicited.

Once an analysis framework was specified, it could be used by practitioners. To do so, the practitioners must first identify scenarios of interest. Following the method, it is necessary to describe each scenario based on the analysis framework as a model instantiating the previously created, included metamodel. Therefore, the requirement "the tool shall support the instantiation of metamodels in terms of models" was identified.

The method suggests that a practitioner will replace the general data with specific information to describe a particular scenario. This results in a more specific description of the considered scenario. This step of the method led to the requirement "the tool shall support the overriding of default values".

In the next step, analysis, the practitioner uses inference to obtain the quantitative values of the models' attributes. The authors of the method suggest inferring values using probabilistic reasoning, which allows uncertainty to be considered. In particular, the authors of [138] have identified five areas of uncertainty: definitional uncertainty, theoretical heterogeneity, causal uncertainty, empirical uncertainty and structural uncertainty. To capture this approach to inference, the requirement "the tool shall support the derivation of attribute values based on probabilistic reasoning" was specified.

Once inference was performed, the authors of the considered method suggest that the inferred results should be visualized. To include this ability, the requirement "the tool shall offer visualization of the calculation results" was identified.

Table 4.3 summarizes the aforementioned requirements.

Requirements derived from the considered Enterprise Architecture analysis method
The tool shall support the creation of extended metamodels describing system properties
The tool shall provide means to express the impact of attributes in terms of calculation rules
The tool shall consider parents of an attribute and the structure of the model during the derivation of attribute values
The tool shall support the specification of default values for attributes
The tool shall support the instantiation of metamodels in terms of models
The tool shall support the overriding of default values
The tool shall support the derivation of attribute values based on probabilistic reasoning
The tool shall offer visualization of the calculation result

Table 4.3: Requirements derived from the considered Enterprise Architecture Analysis method

4.3 Requirements derived from CAD tools

As stated in Chapter 1 for the described research project, current Enterprise Architecture tools should be combined with insights gained from CAD (computer-aided design) tools. Therefore, this type of software tools also needs to be considered when identifying requirements for the presented contribution.

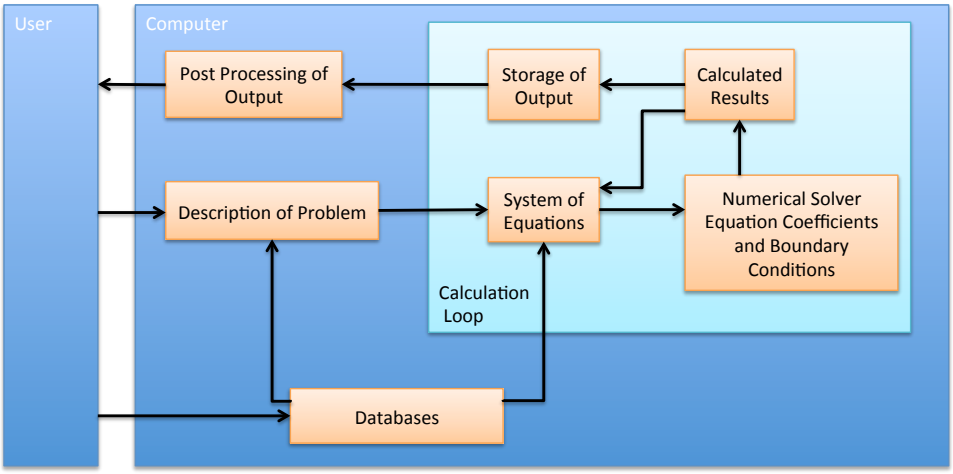
CAD tools assist in the creation, modification, analysis and optimization of a design [179, 162]. There is no generic CAD software covering all possible use cases. Instead, there is a wealth of CAD tools addressing different problems on different levels. The scopes of application include building and architecture, architecture for transportation facilities, product design, automotive engineering, shipbuilding, N-body simulation, dental technology and finite element analysis [162, 68, 278].

As the scopes of CAD tools vary, it is not possible to compare those tools using the same evaluation criteria, nor would it make sense. Someone interested in constructing a technical machine is not interested in urban design and would therefore not benefit from comparing tools for these two different purposes. This leads to the challenge that, unlike TUM and Gartner's Enterprise Architecture tool evaluations, there is no common evaluation criteria catalog available that can be used to directly derive the requirements of a CAD tool for Enterprise Architecture.

One way of identifying requirements indirectly is by considering the general, descriptive literature on CAD tools and using the abstract characteristics of CAD tools as input. [273] sketches the current structure of CAD tools for architecture modeling and analysis (cf. Figure 4.2). In particular, the analysis components of such tools are described. In total, the authors identified eight components: user level, problem description level, systems of equations, numerical solver levels,

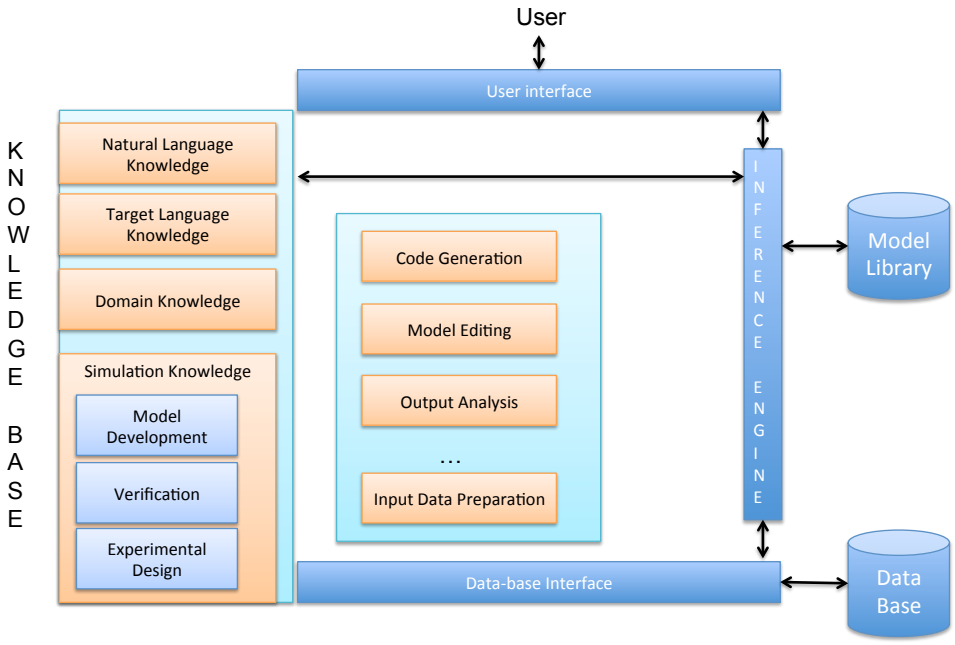
calculated results, storage and post-processing of output, extensions to capability and computing environment.

Figure 4.2: The stages of simulation identified by [273]



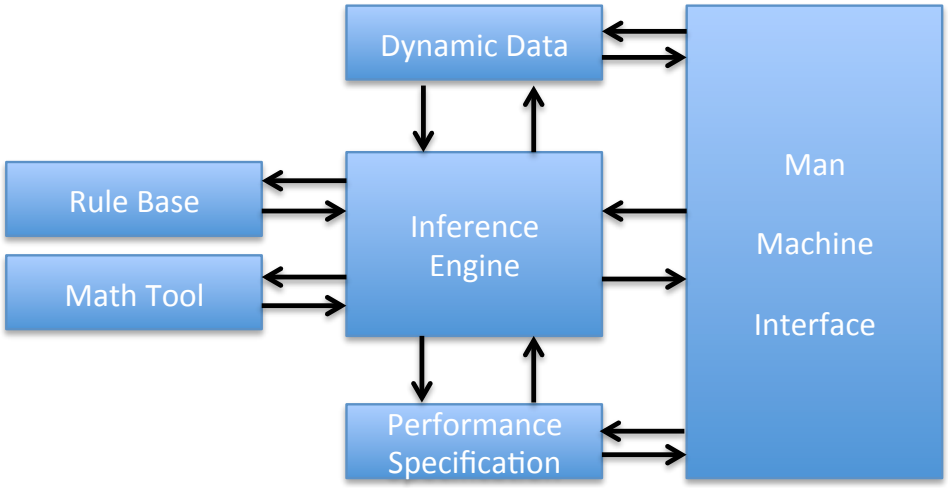
Many CAD tools use expert systems to perform analyses. [229] describes a generic approach for an expert system is described (cf. Figure 4.3). This approach outlines a user interface that is connected to an inference engine. This inference engine is the central component. It is connected to a model library, which is used to store created models at any state of the simulation process. The inference engine also possesses a connection to a database with default values via a database interface. According to the authors, this is a component library allowing the loading of default values. The inference engine is also connected to a knowledge base storing four different types of information: simulation knowledge, domain knowledge, target language knowledge and natural language processing knowledge. Simulation knowledge describes the algorithm used to perform a simulation. Domain knowledge is, according to the authors, the knowledge about the problem. Target knowledge describes the knowledge regarding creating a valid input that can be used to perform simulation upon. Finally, natural language processing knowledge refers to a user interface to the CAD tool that uses natural language to capture the user's input.

Figure 4.3: A combined view of an expert simulation system [229]



[69] is another publication describing the basic components of an expert system (cf. Figure 4.4). The authors identify six components: dynamic data manipulation and creation, performance specifications, a man-machine interface, a rule-base, computational tools and an inference engine.

Figure 4.4: The Expert system structure for computer aided design of feedback controllers [69]



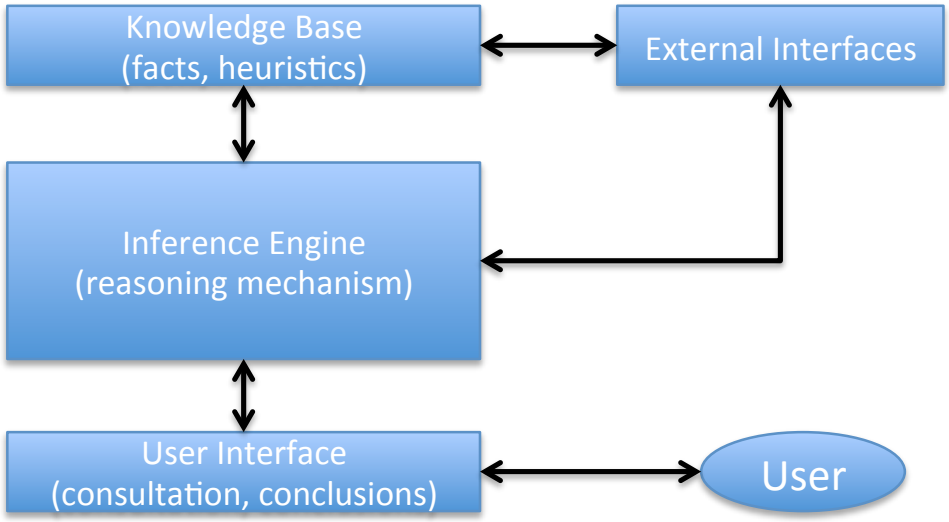
In [171], a summary illustrating the common ground of the described approaches can be found. CAD tools have a user interface supporting the man-machine interaction. This user interface is the connection to the inference engine actually performing the analysis. The inference engine might make use of numerous knowledge bases. These knowledge bases include simulation algorithms, domain knowledge, default values and component libraries as well as information on how to express a problem such that the inference engine can solve it. Additionally, external interfaces are available to connect the knowledge base and inference engine to other tools or components used to visualize results, specify rules for the inference or perform specific mathematical calculations.

Table 4.4 summarizes the requirements derived from a consideration of CAD tools and expert systems.

Requirements derived from CAD tools and expert systems
The tool shall offer a user interface
The tool shall feature a build-in inference engine
The tool shall make use of a knowledge base
The tool shall offer external interfaces

Table 4.4: Requirements derived from CAD tools and expert systems

Figure 4.5: The Expert system's architecture[171]



4.4 Requirements summary

Table 4.2, Table 4.3 and Table 4.4 present requirements from three different domains: Enterprise Architecture tool surveys, a method for Enterprise Architecture analysis and CAD tools and expert systems. The elicited requirements partially overlap or are closely related to one another. This section presents a summary considering this redundancy.

Both the Enterprise Architecture tool evaluations and the requirements derived from CAD tools and expert systems stress the need for the tool to be easy to use. Based on the tool evaluations, the tool should offer a high degree of usability, whereas the CAD tools and expert systems indirectly consider this domain by requiring a user interface. The requirement "the tool shall offer a high degree of usability" was used to cover this requirement.

All three sources of these requirements target the area of analysis. For the tool evaluations, the requirement "the tool shall possess analysis capabilities" was identified. Based on the presented method for Enterprise Architecture analysis, the requirements "the tool shall provide means to express the impact of attributes in terms of calculation rules", "the tool shall support the derivation of attribute values based on probabilistic reasoning" and "the tool shall consider parents of an attribute and the structure of the model during the derivation of attribute values" were identified. Based on the consideration of CAD tools and expert systems, the requirement "the tool shall feature a built-in inference engine" was identified. These requirements were aggregated into the requirement "the tool shall possess analysis

capabilities".

Only the Enterprise Architecture tool evaluations identified the need for administrative capabilities. Therefore, the requirement "the tool shall possess administrative capabilities" was retained.

The Enterprise Architecture tool evaluations stressed the relevance of presentation capabilities. This feature was also recognized as a requirement from the CAD tools and expert systems, again expressed as part of the requirement "the tool shall offer a user interface". Here, the requirement "the tool shall possess presentation capabilities", elicited based on the Enterprise Architecture tool evaluations, was retained.

Another requirement identified by all three sources was the extendable metamodel. Based on the Enterprise Architecture tool evaluations, the requirement "the tool shall support an extendable metamodel" was elicited. The Enterprise Architecture analysis method is more specific. Two requirements addressing extendable metamodels were identified: "The tool shall support the creation of extended metamodels describing system properties" and "the tool shall provide means to express the impact of attributes in terms of calculation rules"

In the field of CAD tools and expert systems, the establishment of a knowledge base is required, which would naturally refer to an extended metamodel in the field of Enterprise Architecture [257]. The requirement "the tool shall support an extendable metamodel" was used to express the various requirements of the metamodel.

Both the Enterprise Architecture tool evaluations and the computer-aided design tool requirements stress the importance of importing, editing and validating capabilities. The requirement "the tool shall support the import, editing and validation of data from external sources" was retained.

The Enterprise Architecture tool evaluations identified the requirement that "the tool shall support the storage of models instantiating a metamodel in a repository". This requirement was retained.

The need for a powerful metamodel was expressed by the Enterprise Architecture tool evaluations. Considering these evaluations, the requirement "the tool shall support the creation of metamodels covering the domains business architecture, information architecture, technology or technical architecture and solution architecture" was elicited. Based on the Enterprise Architecture analysis method, the requirement "the tool shall support the creation of extended metamodels describing system properties" was identified. The wording introduced based on the Enterprise Architecture tool evaluations and the requirement "the tool shall support the creation of metamodels covering the domains business architecture, information architecture, technology or technical architecture and solution architecture" were adopted.

Finally, the Enterprise Architecture tool evaluations describe the need for support for the creation of models. This requirement can also be found in the requirements elicited from the Enterprise Architecture analysis method, wherein more fine-grained versions of the requirements "the tool shall support the instantiation

of metamodels in terms of models" and "the tool shall support the overriding of default values" can be found. "The tool shall support the creation of models" was used as an umbrella term.

The described summary can be found in Table 4.5. Following the research method, these requirements were used to evaluate the tool. This evaluation is presented in Chapter 9.

Summarized Requirements	Including	Source
The tool shall offer a high degree of usability	The tool shall offer a high degree of usability	EATE
	The tool shall offer a user interface	CAD
The tool shall possess analysis capabilities	The tool shall provide support for impact analysis and reporting	EATE
	The tool shall provide means to express the impact of attributes in terms of calculation rules	EAAM
	The tool shall support the derivation of attribute values based on probabilistic reasoning	EAAM
	The tool shall feature a build-in inference engine	CAD
The tool shall possess administrative capabilities	The tool shall possess administrative capabilities	EATE
The tool shall possess presentation capabilities	The tool shall possess presentation capabilities	EATE
	The tool shall provide support for impact analysis and reporting	CAD
The tool shall support an extendable metamodel	The tool shall support an extendable metamodel	EATE
	The tool shall support the creation of extended metamodels describing system properties	EAAM

Continued on next page

Table 4.5 – continued from previous page

Summarized Requirements	Including	Source
	The tool shall support the specification of default values for attributes	EAAM
	The tool shall make use of a knowledge base	CAD
The tool shall support the import, editing and validation of data from external sources	The tool shall support the import, editing and validation of data from external sources	EATE
	The tool shall offer external interfaces	CAD
The tool shall support the storage of models, instantiating a metamodel, in a repository	The tool should support the storage of models instantiating a metamodel in a repository	EATE
The tool shall support the creation of metamodels covering the domains business architecture, information architecture, technology or technical architecture and solution architecture	The tool shall support the creation of metamodels covering the domains business architecture, information architecture, technology or technical architecture and solution architecture.	EATE
	The tool shall support the creation of extended metamodels describing system properties	EAAM
The tool shall support the creation of models	The tool shall support the creation of models	EATE
	The tool shall support the instantiation of metamodels in terms of models	EAAM
	The tool shall support the overriding of default values	EAAM

Continued on next page

Table 4.5 – continued from previous page

Summarized Requirements	Including	Source

Table 4.5: Summarized requirements (EATE: Enterprise Architecture tool evaluations, EAAM: Enterprise Architecture analysis method, CAD: Computer Aided Design Tool)

Chapter 5

Design decisions

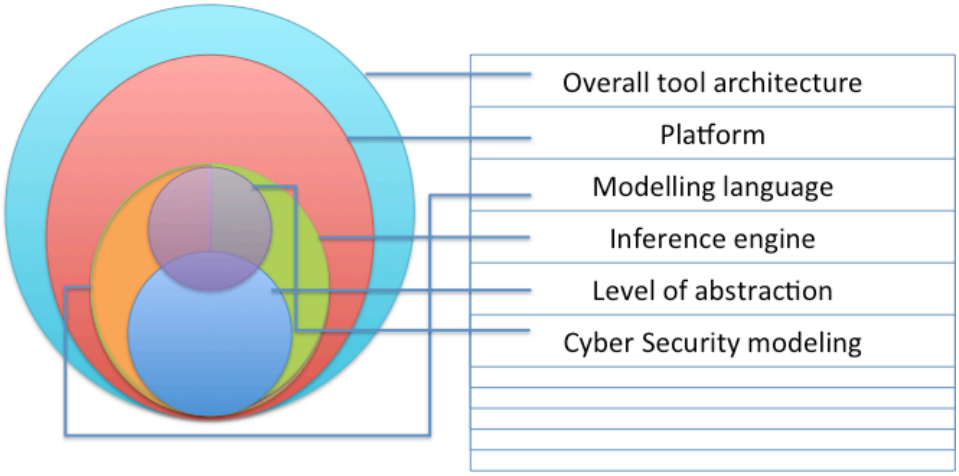
To accomplish the purpose of the research work described in this thesis, several fundamental decisions regarding the design of the artifact, i.e., the tool for Enterprise Architecture analysis, had to be made in a manner in line with the followed research method (cf. Chapter 2).

The areas for decision-making arose iteratively during the design of the tool, which followed the process described in Chapter 6. Once a design decision was made, its outcome generated the need for further decisions. Decisions needed to be made until the design of the artifact was complete, i.e., until the resulting design was considered to be powerful enough that the goal of the research work could be achieved and the identified requirements could be met. Following the used research method, the final result was evaluated against the identified requirements at a later stage of the performed research project. This evaluation is documented in Chapter 9.

At first, a general decision regarding the overall tool architecture needed to be made. Based on this decision, a suitable (rich client) platform could be selected and used to realize the designed architecture. Once a decision regarding the platform was made, a suitable modeling language could be selected. This modeling language was realized based on the capabilities of the platform. The selection of the platform also allowed the inference engine to be chosen, as the platform's features could be used to implement this component. With the decisions regarding the modeling language and inference engine in place, the level of abstraction for the models could be decided, as the level of abstraction is restricted by the modeling language and inference engine's expectation of input in terms of models. Finally, the support for cyber security modeling, a particular work task of the performed research project (cf. Chapter 1), could be realized considering the previous modeling language, inference engine and level of abstraction decisions. The design of the cyber security modeling support required the decisions regarding the modeling language to be made because, to ensure a deep integration, the modeling should be realized with the available modeling language. Again, for reasons of integration,

the cyber security modeling needed to be realized on top of the built-in inference engine, allowing the reuse of this component. The decisions regarding the level of abstraction needed to be considered for the design of the cyber security capabilities, as the level of abstraction determines the character of the provided input used for cyber security analysis. The hierarchy of decisions is visualized in Figure 5.1.

Figure 5.1: The decision hierarchy



The decisions in the previously discussed areas were made under consideration of the requirements stated in (cf. Table 4.5).

In the following, a mapping between the requirements and the areas for decision-making that they affected is established. The relationship between the requirement and the area of design decision-making is discussed briefly. A more extended discussion of the individual relationships is then presented in the following subchapters for each decision area.

There did not exist a 1-to-1 relationship between the requirements and the area for making a design decision that needed to be considered. Instead some requirements impacted the decision-making in several areas and some decisions addressed multiple requirements (n:m relationship). To achieve high usability, the user and how he or she will use the tool need to be considered already at an early stage during the design of the overall architecture and during the decisions involved in this task. Usability can also be supported by the rich client platform used, as such a platform often contains a framework fostering the creation of user-friendly applications. The requirement that the tool offer a high degree of usability also impacts the choice of the modeling language. Some languages might be considered to be more intuitive and thus contribute to a higher usability than others. A selection of the used platform should therefore consider usability aspects as well.

Finally, the level of abstraction impacts how users perceive the created models. A decision regarding the level of abstraction is therefore also impacted by the usability requirement.

The requirement that the tool shall possess analysis capabilities impacts the selection of the inference engine, as this component is responsible for deriving attribute values, an important activity during the performance of Enterprise Architecture analysis. This requirement further impacts the selection of the modeling language. This is the case, as input for the inference engine needs to be specified in some notation in the presented tool using the modeling language.

The requirement that the tool shall possess administrative capabilities also impacted the decision-making regarding the rich client platform used. Some platforms provide framework components fostering the implementation of administrative capabilities.

To address the requirement that the tool shall possess presentation capabilities, this aspect needed to be considered during the selection of the platform and while deciding on the level of abstraction. Some rich client platforms offer features easing the presentation of information. This aspect needed to be considered when deciding on the platform to use for the presented research work. Moreover, the requirements concerning the presentation capabilities impact the decision-making regarding the level of abstraction of the used models, as the demand to present information in a certain way requires that the information be appropriately available for presentation.

The requirement that the tool shall support an extendable metamodel needed to be considered during the decision-making regarding the overall tool architecture, as the support and representation of the metamodel needed to be determined on an overall architecture level to create a sound information architecture. The requirement of supporting an extendable metamodel also impacted the choice of the supported modeling language. In the area of Enterprise Architecture tools, metamodels and modeling language are closely related concepts: modeling languages can be used to express metamodels [232]. Therefore, demanding an extendable metamodel impacts the choice of the modeling language. A selection needed to be made allowing for the extension of the metamodel. Finally, in the context of the presented research work, the inference engine uses models, instantiating extendable metamodels, as input. The decision regarding the inference engine therefore needed to consider their ability to handle extended metamodels and the instantiation thereof.

A consideration of the requirement that the tool shall support the import, editing and validation of data from external sources impacted the choice of the used platform. Some platforms were found to support the import of information based on common, standardized file formats.

The requirement that the tool shall support the storage of models instantiating a metamodel in a repository impacted the overall tool architecture and the modeling language. The decision regarding the overall tool architecture was affected, as the

tool architecture needs to be designed in a way that it offers the storage of models that are created based on metamodels.

The requirement that the tool shall support the creation of metamodels covering the domains business architecture, information architecture, technology or technical architecture and solution architecture needed to be considered during the design of the modeling language and the support for cyber security modeling. The modeling language needed to be selected such that it was capable of expressing the required domains. For cyber security modeling, it needed to be ensured that a consideration of these areas from a cyber security perspective was possible.

Finally, the requirement "the tool shall support the creation of models" impacted three design decisions: the modeling language, the level of abstraction and the support for cyber security modeling. The impact on the modeling language was due to models being created based on a modeling language. The level of abstraction was impacted because models are created at a certain level of detail. The impact on the support of cyber security modeling a result of cyber security models being one form of models.

Table 5.1 describes the previously described mapping between the requirements and the areas for decision- making that they affected

Summarized Requirements	**Impact on decision area**
The tool shall offer a high degree of usability	Design decision I: Overall tool architecture Design decision II: Platform Design decision III: Modeling language Design decision V: Level of abstraction
The tool shall possess analysis capabilities	Design decision III: Modeling language Design decision IV: Inference engine
The tool shall possess administrative capabilities	Design decision II: Platform
The tool shall possess presentation capabilities	Design decision II: Platform Design decision V: Level of abstraction
The tool shall support an extendable metamodel	Design decision I: Overall tool architecture Design decision III: Modeling language Design decision IV: Inference engine
The tool shall support the import, editing and validation of data from external sources	Design decision II: Platform
The tool shall support the storage of models, instantiating a metamodel, in a repository	Design decision I: Overall tool architecture

Continued on next page

Table 5.1 – continued from previous page

Summarized Requirements	Impact on decision area
The tool shall support the creation of metamodels covering the domains business architecture, information architecture, technology or technical architecture and solution architecture	Design decision III: Modeling language Design decision VI: Cyber security modeling
The tool shall support the creation of models	Design decision I: Overall tool architecture Design decision III: Modeling language Design decision V: Level of abstraction Design decision VI: Cyber security modeling

Table 5.1: Mapping between requirements and design decisions

It can be noted that design decisions were made in favor of the overall goal of the performed research i.e., **to develop and demonstrate an Enterprise Architecture modeling tool with a focus on system property analysis** (cf. Chapter 1). Therefore, overall, aspects related to Enterprise Architecture analysis were considered to be more important than general aspects not leading to the fulfillment of the goal of the research documented in this thesis.

5.1 Design option I: Overall tool architecture

The Enterprise Architecture tool resulting from the research work described in this thesis was developed with the goal of advancing the field of Enterprise Architecture tools. The tool should distinguish itself from other available solutions by having a focus on the analysis of Enterprise Architecture models. Compared to other tools, the analysis of system properties should be supported more extensively.

While designing the tool described in this thesis, how the tool's strengths could be reflected in its software architecture had to be determined. A holistic solution that made the features distinguishing the tool from other solutions accessible to its users without overwhelming or confusing them was sought. It was realized that the tool usability could benefit the software architecture developed in accordance with the goals of the research work.

As stated in the previous section, four requirements impacted the design decision regarding the overall tool architecture.

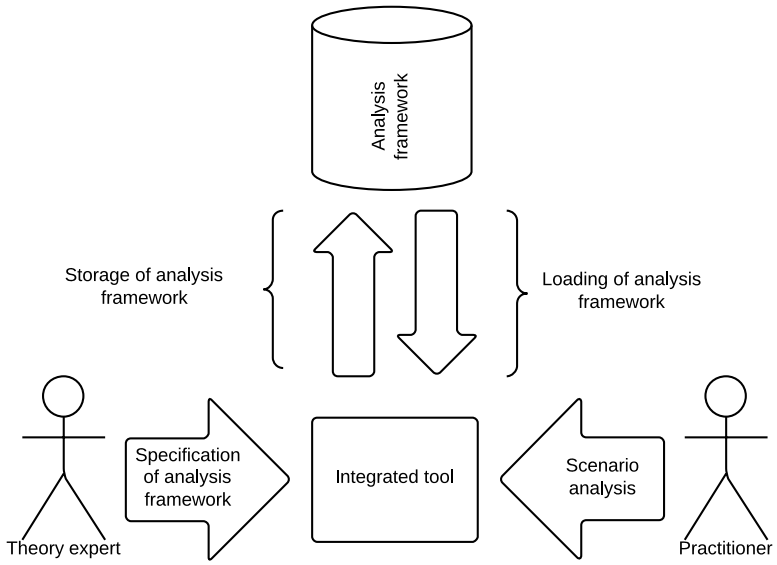
- The tool shall offer a high degree of usability
- The tool shall support an extendable metamodel
- The tool shall support the storage of models, instantiating a metamodel, in a repository
- The tool shall support the creation of models

Moreover, the research project described in this thesis was fairly dynamic with frequently changing requirements. Thus, the support of the tool development was also considered during the evaluation of architecture alternatives.

A structured a workflow was contained in the considered method for the analysis of Enterprise Architecture models (cf. Section 3.2). This workflow suggested the serial working of theory developers and theory applicants. The workflow does not include any collaboration of the two groups in a sense that they work side-by-side. These groups only interact on an administrative level, e.g., when the theory developer educates the theory applicant or when the theory applicant suggests improvements based on his or her experiences gained from using the analysis framework. The architecture of the tool is impacted by the serial order of the performed activities suggested as part of the method. In general, the tool is first used by a theory expert to create an analysis framework, including an extended metamodel. Thereafter, the resulting analysis framework is used by a practitioner. He or she evaluates one or several scenarios in terms of models instantiating the metamodel that is included in the analysis framework. The practitioner may also utilize templates or viewpoints (based on the outcome of design decision V) that are eventually included in the analysis framework and specified on the extended metamodel. These two groups should be supported by the tool and provided with the necessary functionally for performing their particular task to increase the usability of the tool. In addition, the information flow from the theory expert to the practitioner needs to be covered, and the created information needs to be stored in a repository or similar solution.

Three possible ways to provide tailored support for the user were identified. Initially, the implementation of a combined application for both theory experts and theory applicants was considered (cf. Figure 5.2).

Figure 5.2: The simplified integrated architecture



Within the same application, theory experts could specify analysis frameworks and the included extended metamodels and practitioners could create models based on the metamodels included in the analysis frameworks. From a development perspective, this was considered to be an interesting option, as it would ensure a single point of truth, avoiding redundant code [244].

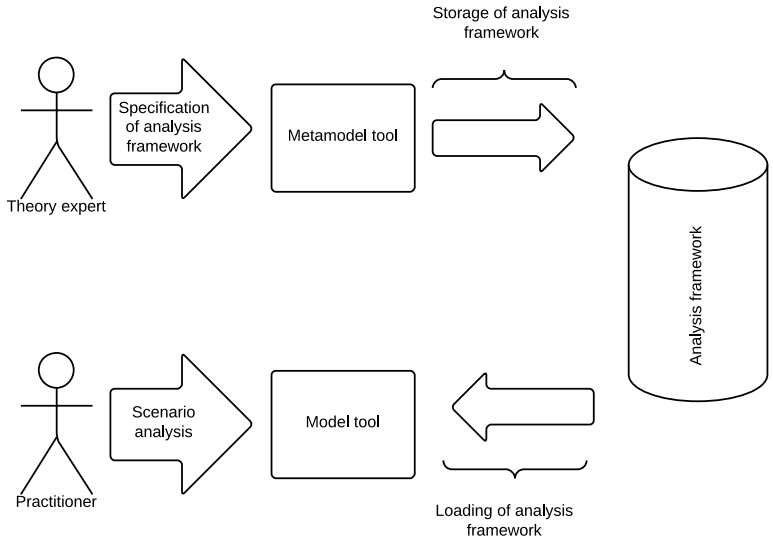
The connection of the metamodel and instantiating model, the most important information objects in the tool, could also be realized fairly easy. These files could be stored in one common file format. A visually homogeneous user interface could also be realized comparably easily in a combined application for theory experts and practitioners.

However, several disadvantages were identified that ultimately led to the elim-

ination of this option. It was realized that the fact that both user groups need different functionalities to perform their tasks leads to an information overflow that negatively impacts the tool's usability. In particular, a negative impact on the usability was expected because a separation of concerns would be difficult to realize. Moreover, it was suspected that an integrated application would have an excessively large code base, negatively impacting the modifiability of the application. The addition of new features or the alteration of existing ones was expected to be difficult due to a complicated and relatively opaque code structure.

Another design option considered was to develop two independent software tools (cf. Figure 5.3).

Figure 5.3: The simplified architecture based on two tools



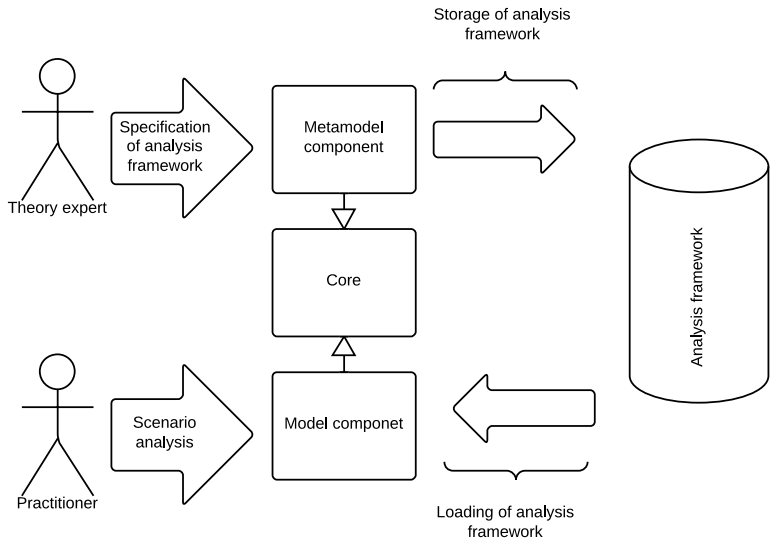
In such a solution, one tool would be developed specifically for the specification of analysis frameworks in terms of extended metamodels. A second, separate tool would allow the creation of models based on those metamodels to evaluate scenarios of interest.

It was expected that a clear separation would improve the usability, as the user interface would be cleaner and only offer the needed functionality. A positive impact on the modifiability of the two software tools was expected, as these would

have a thinner code base, simplifying the implementation of changes. Again, several disadvantages were identified for this design option, and it was ultimately discarded. Compared to the previous alternative, a stronger need for synchronization of the used data structure was identified. In particular, compatibility between the two tools would need to be ensured carefully. If the data structure used in the first component to represent the metamodel were modified, these changes would have to be propagated to the second component to ensure the proper operation of the tool. Another costly disadvantage was identified in the fact that, in some cases, the same tool functionality would have to be developed twice. For instance, the implementation of copy and paste of metamodels elements would have to be redone on a model level. This need was expected to arise for several features of the user interface in particular. Finally, the risk of choosing to develop two applications with a heterogeneous look and feel was realized. Implementing two tools would likely lead to small differences in their user interface, for instance, with regards to the order of items in the menu structure. This inconsistency was considered to cause confusion and thereby negatively impact the usability.

Finally, a common core with two specializations was identified as the third design option (cf. Figure 5.4). This option was chosen after the evaluation of its strength and weaknesses.

Figure 5.4: The simplified architecture based on two tools sharing a core



The two specializations included in this option support the different user groups, and the functionality relevant for both components only needs to be implemented once. One specialization would support the ability of theory experts to specify analysis frameworks consisting of extended metamodels. The other component would support the creation of models based on the metamodels included in the analysis frameworks to apply the frameworks and evaluate scenarios.

In this architecture, the advantages of the two previously descripted design options are combined. First, the development is simplified, as no redundant work needs to be performed; instead, this can be carried out on the core level. A heterogeneous look and feel is created, as a consistent user interface can be realized on the core level as well. Additionally, the information exchange between the two components can be realized within the core.

The storage of models instantiating metamodels could be handled by the core as well. Another identified advantage is the fact that the code base is kept comparably thin, as the core only contains the common code of the tool components and the two components only contain their individual additions. Finally, it was also identified that in such an application, the right information is available at the right moment

and the features relevant for different user groups are only available for them. This was expected to have a positive impact on the usability. This design option is not perfect either. In particular, the design of the common core was recognized as challenging. The proper operation of the two tool components depends on the core, which therefore needs to be designed and implemented carefully. In particular, the core needed to be designed in a flexible way ensuring the tools modifiability.

Table 5.2 compares the discussed tool architectures.

Architecture	**Strength**	**Weakness**
One application for theory specification and theory application	• Support of an extendable metamodel • Support for the creation of model • Support of tool development – Development with a single point of truth • Storage of models, instantiating a metamodel, in a repository – Deep integration of metamodel and model • Contributes to a high level of usability – Homogeneous user interface	• Negative impact on usability – No separation of concern – Overwhelming information • Negative impact on tool development – One large code base – Low modifiability

Continued on next page

Table 5.2 – continued from previous page

Architecture	Strength	Weakness
Two applications, one for theory experts and one for theory applicants	• Support of an extendable metamodel • Support for the creation of model • Contributes to a high level of usability – The right information at the right moment for the different user groups – Clean user interface • Support of tool development – High modifiability – Thin code base	• Negative impact on tool development – Stronger importance of synchronized information between metamodels and models – Compatibility issues – Same functionality developed twice • Negative impact on usability – Eventually heterogeneous look and feel user interface

Continued on next page

Table 5.2 – continued from previous page

Architecture	Strength	Weakness
A common core and two specializations	• Support of an extendable metamodel • Support for the creation of model • Support of tool development – Simplified development as no redundant work needs to be done – Thinner code base compared to one integrated application • Contributes to a high level of usability – Heterogeneous user interface – The right information at the right moment for the different user groups – Simplified information exchange • Storage of models, instantiating a metamodel, in a repository	• Negative impact on tool development – Whole application depends on the common core

Table 5.2: Comparison of the potential architecture candidates

5.2 Design option II: Platform

Currently, the development of a software tool is typically performed based on the usage of one or several (rich client) platforms. These platforms offer standard components, including menus, toolbars, internationalization and help systems, that are part of almost any application [46]. Reimplementing these components is expensive, prone to introducing errors and results in a heterogeneous application design. Instead, the user feels easily comfortable when the same building blocks are reused and the same patterns are followed across different applications.

As described in the introductory section of this chapter, the platform selection was impacted by the following requirements.

- The tool shall offer a high degree of usability
- The tool shall possess administrative capabilities
- The tool shall possess presentation capabilities
- The tool shall support the import, editing and validation of data from external sources

Another factor that needed to be considered during the selection of the platform was support for the tool development. For a dynamic project such as the research project described in this thesis, a lively user community is of great importance.

The components that are included in a rich client platform also support the administrative capabilities of a tool, i.e., the tool's capability to handle large-scale models. This is the case because these platforms are optimized to handle large amounts of data.

Furthermore, using the included components of rich client platforms, one can design user interfaces that support the presentation of information. In the context of the discussed research, the capability to present information can be utilized to visualize models or parts thereof.

Finally, rich client platforms sometimes foster the realization of import and export functionality, as they provide components that allow the reading and writing of files following commonly used file standards. Editing and validating such imports and exports is also possible. Thus, it was decided that an existing rich client platform would be used for the realization of the previously designed overall architecture. The considered platforms had to be based on Java, as this was the best-understood language of the project team. Moreover, it had to be open-source for financial and political reasons. A focus on desktop applications was another choice, as all existing Enterprise Architecture tools are available for this architecture platform as well [166]. The final choice was support for the creation of modeling applications as a sine qua non for the implementation of an Enterprise Architecture tool.

These requirements reduced the list of possible candidates down to two: NetBeans [30, 200] and the Eclipse Rich Client Platform[167, 226]. Yet again, the selection was made following the iterative method described in Chapter 2.

With the NetBeans Visual Library [202], NetBeans offers a Java-based component library for the creation of modeling tools. The usage of this library is fairly

easy, and it was applied during the first versions of the tool [44]. In particular, the included Matisse GUI Builder [201] simplified the creation of the user interface.

However, as the tool grew larger during development, several bugs and weaknesses of the NetBeans Visual Library were identified and reported in the official forums. These weaknesses were so drastic that an initiative to identify alternatives was initiated, including an evaluation of the Eclipse Rich Client Platform as a master thesis [127].

A change in the ownership of Sun Microsystems [197], the company behind NetBeans, also affected NetBeans itself. Oracle, the new owner of NetBeans, offered two Java development environments: the proprietary JDeveloper [203] and the open-source NetBeans. As a result of this dualism, the development team of the presented Enterprise Architecture analysis tool reported a decline in the support for NetBeans. In particular, reported bugs were not solved as desired, and the Visual Library was not updated at all.

Thus, based on the outcome of [127] the decision was made to phase out NetBeans and to instead use the Eclipse Rich Client Platform. Eclipse not only has a very active community but also provides a significantly larger number of reusable components [268], plugins in the Eclipse jargon, that help to easily create advanced applications. In particular, the Eclipse Modeling Framework (EMF) [250, 76] was useful for the development of the presented tool. It was used during the implementation of both the user interface and the inference engine. However, a discussion of the architecture of the tool and usage of the EMF is beyond the scope of this section. Instead, the reader is referred to Chapter 7, which elaborates on the tool architecture.

The Eclipse Rich Client platform has a lively and supportive user community that supports the tool development. In terms of weaknesses, this platform lacks a no non-commercial tool for the creation of a graphical user interfaces.

Table 5.3 summarizes the described platform comparison.

Platform	Strength	Weakness
Eclipse Rich Client Platform	• Contributes to a high level of usability – Integration with the operating system • Supports realizing administrative capabilities – Using existing plugins for such tasks • Supports realizing presentation capabilities – Using existing plugins for such tasks • Supports realizing import, edit and validate data from external sources – Using existing plugins for such tasks • Support of tool development – Good community support – Powerful platform – Many projects developing plugins that can be integrated in other applications – Commonly used	• Negative impact on tool development – No graphical GUI builder available

Continued on next page

Table 5.3 – continued from previous page

Platform	Strength	Weakness
NetBeans	• Contributes to a high level of usability – Integration with the operating system – Matisse GUI Builder • Supports realizing administrative capabilities – Using included components • Supports realizing presentation capabilities – Using included components • Supports realizing import, edit and validate data from external sources – Using included components • Support of tool development – Easy to use	• Negative impact on tool development – Community comparably less responsive – Development of the Rich Client Platform less lively

Table 5.3: Comparison of the potential rich client platforms candidates

5.3 Design option III: Modeling language

To develop a tool for Enterprise Architecture analysis, a modeling language must be chosen to give the tool user a ready notation that he or she can use to create input that then can be processed by the software tool. The modeling language needed to be selected with the goal of the presented research work in mind. In the first part of this chapter, the requirements of a tool for Enterprise Architecture analysis that impact the selection of the modeling language were discussed. These requirements are as follows:

- The tool shall offer a high degree of usability
- The tool shall possess analysis capabilities
- The tool shall support an extendable metamodel
- The tool shall support the creation of metamodels covering the domains business architecture, information architecture, technology or technical architecture and solution architecture
- The tool shall support the creation of models

Potential modeling languages were evaluated against these requirements.

In the field of Enterprise Architecture, especially the academic community performing research in that field, ArchiMate [153] is widely used. ArchiMate offers a defined metamodel that covers business architecture, information architecture, technology architecture and solution architecture (using the Motivation extension [66]). It is considered to be fairly easy to understand and therefore possesses a high usability.

[153] states that TOGAF is the most popular Enterprise Architecture framework. Since the release of ArchiMate 2.0 in 2012, ArchiMate completely supports TOGAF [103]. It is reasonable to assume that ArchiMate will increase in popularity, even outside the academic community. A natural first choice would therefore be to select ArchiMate as the supported modeling language for the tool presented in this thesis.

However, even though ArchiMate is widely used, it has several disadvantages that led to the selection of another language as the default modeling language. ArchiMate has a fixed metamodel defining a number of classes that can be used to describe an enterprise. These classes cover the general aspects of a company; however, some concepts for system property analysis are not included. For instance, neither languages and protocols that are relevant for interoperability analysis [260] nor network interfaces and attack steps that are of importance for cyber security analysis [246] are included. Although ArchiMate is (too) specific concerning some aspects of its metamodel, there are other dimensions in which it is not precise enough. ArchiMate does not specify multiplicities for the defined relations [22], making its usage sometimes ambiguous and negatively impacting the usability. Another drawback is that ArchiMate has no unified support for adding attributes to its metamodel. This limits the extendibility of ArchiMates' metamodel, as there is no standard procedure of how to implement an extension. Limited support for attributes also impacts their consideration if they are added to a model. ArchiMate

does not foresee a standard way to calculate values for modeled attributes.

Nevertheless, several successful attempts have been made, including [170, 123], during which a variety of system properties have been evaluated. However, these attempts all featured an external calculation mechanism, such as queuing theory in the case of [123] or probabilistic relational models in the case of [144] and [185]. Offering a unified method of describing and evaluating the described system properties would, on the one hand, increase the usability of the modeling language, as the models would be easier to understand and, on the other hand, allow to combine different system property analyses and make the results easier to compare. Finally, ArchiMate does not allow the existence or non-existence of described concepts and relationships to be explicitly considered. It is not intended to describe classes or relationships between them that eventually do not exist. Instead, ArchiMate assumes that everything that is part of a created model actually is in place. This makes it difficult, if not impossible, to express structural uncertainty. However, this is an important characteristic that needs to be considered during Enterprise Architecture analysis [138].

The DEMO methodology [219, 61] is a flowchart method that includes a social emphasis. Using DEMO, models can be created that describe actions that take place at organizations in the context of their actors and intended audience [16]. Demo suggests the usage of transactions to describe business aspects of enterprises. Transactions are patterns consisting of interactions and actions. An action is the core of a business transaction, describing an activity that generates a new result. Actors coordinate actions based on interactions, or acts of communication. Business transactions are carried out in three distinct phases: order, execution and result phase. The DEMO methodology makes use of Petri nets to allow the analysis of models created following this approach. The usage of Petri nets allows probabilistic reasoning [147] to be expressed as part of the DEMO methodology.

However, several weaknesses led to the selection of a different methodology. Its strong focus on business aspects, especially business process modeling, was considered to be a disadvantage within the context of Enterprise Architecture. Other domains, such as information architecture, technology or technical architecture and solution architecture, cannot be captured sufficiently. Additionally, the usage of transactions, implying the consideration of three states (before, during and after), made the performance of an activity too specific for the research work described in the thesis. Such system properties as modifiability [149] or availability [182] do not consider enterprises from such a time perspective. Finally, the DEMO methodology does not allow structural uncertainty to be considered.

Frank et al.[78] propose Score-ML, a language for defining (business) indicator systems within enterprise models. These models focus on the business architecture. However, the information architecture, technology or technical architecture and solution architecture of a considered organization can be described to a certain extent as well. Score-ML instantiates the MEMO metamodeling language (MML) [77]. Thus, Score-ML features classic capabilities of object-oriented metamodeling facilities, including cardinalities and primitive data types. Score-ML's indicators

can also consider specialized relationship types, allowing dependencies between different indicators and between an indicator and a goal to be described. Score-ML does not place particular emphasis on the notion of uncertainty, nor does it facilitate the introduction of distributions acting as surrogates for actual indicator values [36]. In the context of the described research project, its analysis capabilities are therefore not sufficient.

Other well-known Enterprise Architecture modeling languages and frameworks, including PEAF, the Pragmatic Enterprise Architecture Framework [243], and FEA, the U.S. Federal Enterprise Architecture Framework [54], have some or all of the disadvantages that led to the removal of the previously discussed languages from consideration. Typically, the metamodel does not cover all of the relevant concepts that are needed for the performance of Enterprise Architecture analysis. A common weakness from an analysis perspective is also the limited ability to consider uncertainty. After considering the common Enterprise Architecture modeling languages and realizing their insufficient support for analysis, the need for a more powerful language was identified.

A more general approach is Alloy [129], a language that captures the essence of software abstractions simply and succinctly, with an analysis that is fully automatic, and that can expose the subtlest of flaws. Using Alloy, one can fully automatically perform semantic analysis as well as check consequences and consistency and simulate execution [128]. Alloy is fairly easy to understand and use [112], which positively contributes to its usability. It allows the creation of models covering the domains of business architecture, information architecture, technology or technical architecture and solution architecture. However, Alloy comes with the disadvantage of not having such notions as field, method or integer arithmetic, which are imperative for the performance of Enterprise Architecture analysis following the method presented in Section 3.2. It also lacks recursively defined constraints, which are important, for instance, in the case of cyber security analyses. Moreover, neither the sequencing of operations nor the use of higher-order quantifiers is possible [128].

The most obvious candidate for use as a modeling language in the presented tool was the UML [102]. Its sheer popularity [240] made it worth considering, as did the fact that many existing (classical) Enterprise Architecture tools including Sparx Systems Enterprise Architect [255] and Mega System Blueprint [125] offer support for the Unified Modeling Language were relevant arguments. Users of those tools can switch to another tool offering the same modeling language fairly easily, contributing to the fulfillment of the requirement that the tool offer a high degree of usability.

An evaluation of the UML revealed that this language was indeed suitable for an Enterprise Architecture analysis tool, leading to its selection. Using UML Class and Objects [102], one can specify Enterprise Architecture metamodels and their instantiations in terms of architecture models. These architecture models can describe such aspects as the business architecture, information architecture, technology or technical architecture and solution architecture of an organization.

Moreover, it is possible to specify languages, protocols, network interfaces, attack steps and a variety of other concepts that were not supported by ArchiMate and the other considered Enterprise Architecture modeling languages. This contributes to the fulfillment of the requirement that the tool support an extendable metamodel. Additionally, with OCL [267], the UML offers a second mechanism to define constraints on the models' structure. OCL can also be used to evaluate the attributes that are included in models created using the UML [5]. Therefore, using the UML contributes to the fulfillment of the requirement that the tool possess analysis capabilities. Another weakness of ArchiMate and similar languages can be addressed. Class diagrams and object diagrams may make use of association classes [102] according to the UML. Using association classes, one can assign attributes to relationships that are included in the class and object diagrams [94]. In this way, it was possible to overcome the final identified weakness of ArchiMate and the other considered modeling languages: the lack of support for structural uncertainty. A mechanism for the specification of structural uncertainty on both the class and the relationship level could be achieved by equipping both concepts with an additional existence attribute (cf. Section 5.4). A weakness of the UML is that it is very powerful and requires some effort to learn.

Table 5.4 summarizes the previously discussed modeling languages and their identified strengths and weaknesses.

Language	**Strength**	**Weakness**
ArchiMate	• Supports the creation of models • Contributes to a high level of usability – Commonly used within the field of Enterprise Architecture • Covers the domains business architecture, information architecture, technology or technical architecture and solution architecture	• Negatively impacts usability – Ambiguities in metamodel • Insufficient analysis capabilities – No defined approach for the evaluation of attributes is suggested • Does not offer a sufficiently extendable metamodel

Continued on next page

Table 5.4 – continued from previous page

Language	Strength	Weakness
DEMO	• Supports the creation of models • Possesses analysis capabilities – Probabilistic reasoning using Petri nets	• Limited analysis capabilities – Focus on transactions – Structural uncertainty not considered • Difficult to model domains information architecture, technology or technical architecture and solution architecture – Focus on business aspects • Does not offer a sufficiently extendable metamodel
Score-ML	• Supports the creation of models • Possesses analysis capabilities – Indicators of architecture elements • Covers the domains business architecture, information architecture, technology or technical architecture and solution architecture	• Does not offer a sufficiently extendable metamodel – Fixed metamodel • Limited analysis capabilities – Capturing of uncertainty not supported

Continued on next page

Table 5.4 – continued from previous page

Language	Strength	Weakness
Alloy	• Supports the creation of models • Contributes to a high level of usability – Is easy to understand • Possesses analysis capabilities – Performance of semantic analysis • Possesses an extendable metamodel • Can cover the domains business architecture, information architecture, technology or technical architecture and solution architecture	• Limited analysis capabilities – No notions as field, method or integer arithmetic – No recursively defined constraints – No sequencing of operations. – No higher-order quantifier

Continued on next page

Table 5.4 – continued from previous page

Language	Strength	Weakness
UML	• Supports the creation of models • Contributes to a high level of usability – Commonly used – Powerful language • Possesses analysis capabilities – Querying using OCL • Possesses an extendable metamodel – Metamodel can be specified as classes and objects • Can cover the domains business architecture, information architecture, technology or technical architecture and solution architecture	• Negatively impacts usability – Can be considered to be too powerful and difficult to learn

Table 5.4: Comparison of potential modeling languages

The following section discusses both UML and OCL and presents relevant background information. Chapter 8 reports on the usage of the tool in general and on the definition of Enterprise Architecture metamodels in terms of UML class diagrams in particular.

UML and OCL

The Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing, documenting and analyzing artifacts. Over the last decade, UML has become the de facto standard for creating models of software-based systems, as well as for modeling business and similar processes [99].

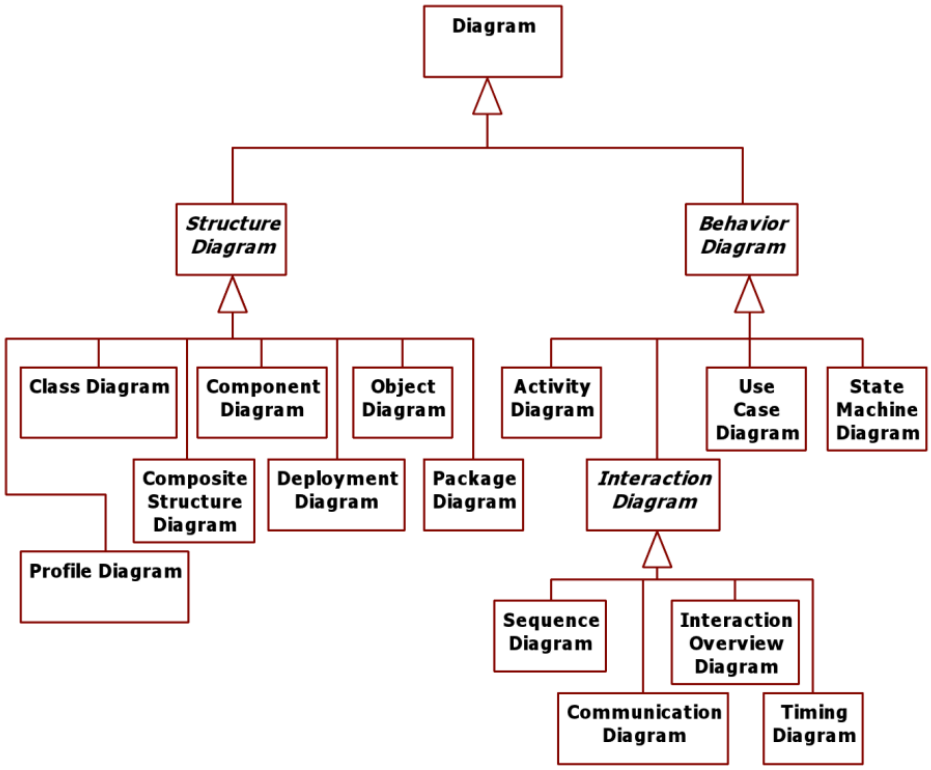
Using UML, one can write system blueprints describing conceptual aspects, business processes and system functions as well as concrete details, such as programming language statements, database schemas and reusable (software) components [99].

The first version of UML was the result of a synchronization of three popular object-oriented methods: Object Modeling Technique (OMT) [228] Object-Oriented Software Engineering (OOSE) [130] and Booch [28]. With revision 2.0, UML [102] has been enhanced with more specific definitions of its abstract syntax and semantics, a more modular structure and better support for the modeling of large-scale systems.

UML's metamodel specifying its abstract syntax is based on MOF [101], the Object Management Group's Meta-Object Facility, a semiformal approach to writing models and metamodels.

UML's abstract syntax specifies the set of modeling concepts, including their attributes and relationships. The rules for combining these concepts to construct partial or complete UML models are covered as well. In total, UML specifies 14 different types of diagrams [17](cf. Figure 5.5). These can be generally divided into two groups. On the one hand one can find structural diagrams describing aspects that must be present in the modeled system. On the other hand there are behavior diagrams explaining what must happen in the described system. The supported structural diagrams are class diagrams, component diagrams, composite structure diagrams, deployment diagrams, object diagrams, package diagrams and profile diagrams. The behavior diagrams specified by UML are activity diagrams, communication diagrams, interaction overview diagrams, sequence diagrams, state machine diagrams and timing diagrams. In the context of Enterprise Architecture modeling, class diagrams and object diagrams are typically of relevance. Class diagrams describe the structure of a system by showing its classes, their attributes and the relationships among the classes. Object diagrams follow class diagrams and are used to illustrate the state of a modeled system at a particular time [102].

Figure 5.5: The 14 UML diagrams, as presented in [17]



It can be difficult to synchronize information over different UML diagrams. The authors of [220] stress that "for certain aspects of a design, diagrams often do not provide the level of conciseness and expressiveness that a textual language can offer". To overcome this weakness, the Object Management Group complemented UML with OCL, the Object Constraint Language [100]. Jos Warmer and Anneke Klepper developed this language based on Steve Cook and John Daniels's Syntropy language [251].

OCL's syntax is similar to common object-oriented languages and query languages, such as SQL. OCL is declarative and can be used to specify constraints on a conceptual level, abstracting them from lower-level implementation details [220]. In particular, it allows the specification of constraints on UML class diagrams. These constraints can then be investigated on the UML object diagrams that instantiate the class diagrams. The consideration of OCL invariants during the instantiation ensures the creation of compliant object diagrams.

OCL expressions are side-effect-free, meaning that the state of the system will

not change based on the result of an OCL expression. OCL is typed; thus, every expression has a type, and all types must match so that the expression is correct [258].

OCL expressions are specified in the context of a model and do not exist without one. These expressions can consider both the structure of the model and the modeled characteristics, i.e., attribute values or states of the model elements. Although OCL was originally mainly created as a constraint language to ensure consistency over UML diagrams, it can also be used as a query language due to its ability to navigate the model and form collections of objects [5]. The selected platform, the Eclipse Rich Client Platform, eased the support of the class and object diagrams by offering the Eclipse Modeling Framework [250], a component that supports modeling according to UML.

5.4 Design option IV: Inference engine

Another important decision that had to be made during the tool implementation was the selection of the used analysis engine. This component is used to evaluate an Enterprise Architecture model. The task of the engine is to infer the values of the attributes that are part of an architecture model given values for a subset of the contained attributes.

The inference engine needed to be selected with the goal of the presented research work in mind. In the first part of this chapter, the requirements of a tool for Enterprise Architecture analysis that impact the selection of the inference engine were listed as follows:

- The tool shall possess analysis capabilities
- The tool shall support an extendable metamodel

In accordance with the iterative character of the method used to perform this thesis, the decision had to be made several times. Typically, the identification of weaknesses of one inference engine led to a consideration of possible alternatives. The most promising alternative was then implemented in the tool after an evaluation.

The first considered inference engine was built on Dempster-Shafer theory [238, 274]. Dempster-Shafer theory allows architectural analysis, including inference of attribute values based on other values, incomplete Enterprise Architecture models and credibility assessments with regards to the provided input [139]. Dempster-Shafer theory is specifically designed to allow the representation of ignorance [137]. However, it lacks support for decision and goal representation. It also does not provide support for managing multiple levels of abstraction or defining new concepts. Compared to class and object diagrams featured by UML, separating analysis theory and application of theory is difficult.

To overcome the weaknesses of Dempster-Shafer theory, a second inference engine based on extended influence diagrams (EIDs) [150] was designed. This engine had stronger analysis capabilities and allowed the expression of decisions and goals. Based on Bayesian probability theory, it also allowed uncertainty to be considered to a certain extent, mainly in the forms of definitional uncertainty, theoretical heterogeneity, causal uncertainty and empirical uncertainty could be covered. However, the issue of missing support for multiple levels of abstraction remained. Therefore, the requirement “the tool shall support an extendable metamodel” could not sufficiently be met by this inference engine.

The third engine was built using probabilistic relational models (PRMs) [143]. A tool version using this engine was presented in [44]. PRMs allowed the separation of theory specification in terms of a class diagram and the application of the theory in an object diagram. In this way, the weakness of the two previous inference engines, which did not sufficiently support extended metamodels, was overcome. PRMs contributed to the analysis capabilities of the tool, as PRMs allowed the modeling of discrete Bayesian attributes. Dependencies between these attributes could be

considered and specified in terms of probability tables. Additionally, default values for the states of the included attributes could be specified.

This engine was later extended to support hybrid probabilistic relational models (HPRMs) [181]. Compared to PRMs, these models allowed the consideration of continuous attributes, i.e., attributes that are not characterized by a defined number of states but by a numerical value. Thus, the analysis capabilities increased. While using this fourth engine, the need for structural analysis arose [38, 259]. For the analysis of some system properties, especially interoperability according to [260], it is important to consider the structure of the model, i.e., to investigate whether certain relationships exist. PRMs and their extension HPRMs, however, were not able to sufficiently support structural analysis.

This led to the implementation of the fifth inference engine, that which is used in the latest version of the tool [43]. This engine uses the Predictive, Probabilistic Architecture Modeling Framework (P2AMF) [140](cf. Section 5.4) to evaluate Enterprise Architecture models. The engine provides the tool with advanced analysis capabilities. P2AMF allows the attribute values to be calculated based on the values of the parental attributes. For attribute values, it is possible to specify default values. Using P2AMF, both continuous and discrete attributes can be considered. It also allows the structural aspects to be considered when evaluating models. Using P2AMF's existence attribute, it is possible to describe structural uncertainty. As P2AMF supports the analysis of models using Markov chains, even definitional uncertainty, theoretical heterogeneity, causal uncertainty and empirical uncertainty can be considered.

P2AMF is designed on top of OCL, which in turn is closely related to UML class and object diagrams (cf. Section 5.3) and allows likewise a separation between the definition of a theory and the application of the defined theory. This characteristic helped address the requirement "the tool shall support an extendable metamodel". The selected rich client platform facilitated the implementation of P2AMF, as it offered components for the evaluation of OCL queries on UML models. However, as the application of P2AMF is not very common, its usage might initially be considered complicated.

Table 5.5 summarizes the discussed inference engines and their weaknesses.

Inference engine	Strength	Weakness
Dempster-Shafer theory	• Analysis capabilities – Calculation of attribute values based on other included attributes	• Insufficient analysis capabilities – Lacking support for decision and goal representation • Insufficient support for extendable metamodels – No separation between metamodel and model
Extendend Influence Diagrams	• Analysis capabilities – Calculation of attribute values based on other included attributes – Decisions and goals can be modeled	• Insufficient support for extendable metamodels – No separation between metamodel and model • Insufficient analysis capabilities – Difficult to consider structural aspects (which had to be hard-coded)

Continued on next page

Table 5.5 – continued from previous page

Inference engine	Strength	Weakness
Probabilistic Relational Models	• Analysis capabilities – Calculation of attribute values based on other included attributes – Possible to specify default values for attributes • Separation between metamodel and model	• Insufficient analysis capabilities – Difficult to consider structural aspects (which had to be hard-coded) – Only discrete attributes can be considered
Hybrid Probabilistic Relational Models	• Analysis capabilities – Calculation of attribute values based on other included attributes – Possible to specify default values for attributes – Consideration of discrete and continuous attributes • Separation between metamodel and model	• Insufficient support of structural analysis

Continued on next page

Table 5.5 – continued from previous page

Inference engine	Strength	Weakness
Predictive, Probabilistic Architecture Modeling Framework	• Analysis capabilities – Calculation of attribute values based on other included attributes – Possible to specify default values for attributes – Consideration of discrete and continuous attributes – Support for structural analysis • Separation between metamodel and model	• Steep learning curve

Table 5.5: Comparison of the used inference engines

Predictive, Probabilistic Architecture Modeling Framework

The Predictive, Probabilistic Architecture Modeling Framework (P2AMF) [140] is an extension of OCL for probabilistic analysis and prediction of system properties. It makes use of OCL's capability to act as a query language. The main feature of P2AMF is its ability to express uncertainties of objects, relations and attributes in UML models and perform probabilistic assessments incorporating these uncertainties. A typical usage of P2AMF would be to create a model for predicting, e.g., the availability of an application. In P2AMF, two types of uncertainty are introduced. First, attributes may be stochastic. When attributes are instantiated, their values are expressed as probability distributions. Second, the existence of objects and relationships may be uncertain. It may be the case that one no longer knows whether a specific server is still in service, which is a case of object existence uncertainty. Such uncertainty is specified using an existence attribute E that is mandatory for all classes (here, using the concept class in the regular object-oriented aspect of the word), where the probability distribution of the instance myServer.E might be P(myServer.E)=0.8

i.e., there is an 80% chance that myServer still exists. It might also be uncertain whether myServer is still serving a specific application, i.e., whether there is a connection between the server and the application. Similarly, relationship uncertainty

is specified with an existence attribute E on the relationships.

The probabilistic aspects are considered in a Monte Carlo fashion. In each iteration, the stochastic variables are instantiated with instance values according to their respective distributions. This includes the existence of classes and relationships, which are sometimes instantiated, sometimes not, depending on the distribution. Then, each of the P2AMF statements is transformed into a proper OCL statement and can be evaluated. How this is realized is described in the following section.

Sampling algorithms

This section explains how inference is performed in the tool. The authors of P2AMF are not specific about how this should be realized. In the tool, three sampling algorithms are implemented to infer the values of the attributes that are part of the created model: forward sampling, rejection sampling and Metropolis-Hastings sampling [161]. Each algorithm has advantages and disadvantages. The Gibbs sampler [49], an alternative algorithm that is also frequently used, was not directly supported, as it is a special case of the more general Metropolis-Hastings algorithm.

The user starts the sampling functionality as soon as the object diagram describes the scenario of interest. The P2AMF object diagram is sampled to create a set of deterministic object diagrams. This is performed considering the probability that a certain object is part of the created object diagram and that a given relationship is contained in that object diagram as well. Therefore, the existence attribute E, as explained above, is evaluated.

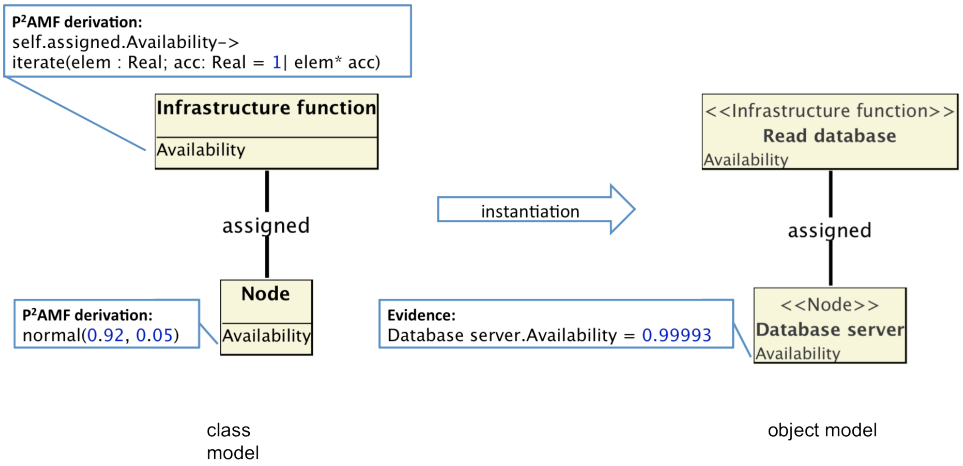
For each of these sample models, standard OCL inference is performed, thus generating sample values for all modeled attributes. For each attribute, the sample set collected from all sampled OCL models is used to characterize the posterior distribution.

For all sampling algorithms, the first step is to generate random samples from the existence attributes' probability distribution $P(X) : x_1, \ldots, x_M$. For each sample, x_i, and based on the P2AMF object diagram Op, a reduced object diagram, $N_i \in N$, containing only those objects and links whose existence attributes, Xj, were assigned the value true is created. Some object diagrams generated in this manner will not conform to the constraints of UML. In particular, object diagrams may appear such that a link is connected to only one or even zero objects. Such samples are rejected. Other generated object diagrams will violate, e.g., the multiplicity constraints of the class diagram. Such samples are also rejected. Additionally, some OCL derivations are undefined for certain object diagrams, for instance, a summation derivation over an empty set of attributes. A set of traditional UML/OCL object diagrams remains with $\Xi \subset N$, whose structures vary but are syntactically correct and whose attributes are not yet assigned values. Finally, if the user provides evidence for one or several attributes, the sample is assigned the evidence value.

Forward sampling

Forward sampling (cf. Algorithm 1) consists of only a few steps and leads to a fast sampling process. However, forward sampling has the disadvantage of not allowing the specification of evidence; on any arbitrary attribute in the object diagrams, only evidence on attributes not calculated based on other attributes' values is allowed.

Figure 5.6: An example application of P2AMF



In the example depicted in Figure 5.6, only evidence for the attribute database server.availability can be provided. Forward sampling requires the attributes that are part of a sample $Y_1, \ldots, Y_n$ to be sorted in topological order, such that parent attributes appear earlier in the sequence than the attributes that are calculated based on them, their children. Thus, *Database server.availability* comes before *Read database.availability* in the example of Figure 5.6. Following the general first step, as described above, the second step of the forward sampling algorithm is that for each of the remaining object diagrams, Ξ_i, the probability distribution of the attributes not calculated based on the value(s) of other attributes, $P(\mathbf{Y}^r)$ is sampled. This creates the sample set $\mathbf{y}^r[1], \ldots, \mathbf{y}^r[size(\boldsymbol{\Xi})]$. If there is evidence on a root attribute, the sample is assigned the evidence value. Based on the samples of the root attributes, the OCL derivations are calculated in topological order for each remaining attribute in the object diagram, $y_i^{\overline{r}} = f_{y_i^{\overline{r}}}(\mathbf{Pa}_{y_i^{\overline{r}}})$. The result is a set of deterministic UML/OCL object diagrams, $\boldsymbol{\Lambda} \subset \boldsymbol{\Xi}$, where in each model, all attributes are assigned values. The final set of object diagrams, $\mathbf{O} \subset \boldsymbol{\Lambda}$, contains attribute samples from the posterior probability distribution $P(\mathbf{X}, \mathbf{Y}|\mathbf{e})$. These samples may thus be used to approximate the posterior.

Algorithm 1: Forward sampling

```
1   for (int i=1; i<M; i++) {
2       x=sampleExistenceAttributes();
3       N = extractObjectModel(Op, x);
4       if (syntacticallyCorrect(N)) {
5           for(int j=1; j<N; j++) {
6           Y_j = sample(getParents(Y_j));
7           Λ = assignAttributesToModel(y, N);
8           O.add(Λ);
9       }
10  }
```

Rejection sampling

The objective of rejection sampling (cf. Algorithm 2) is to generate samples from the posterior probability distribution $P(\mathbf{X}, \mathbf{Y}|\mathbf{e})$, where $\mathbf{e} = \mathbf{e}^X \cup \mathbf{e}^Y$ denotes the evidence of existence attributes as well as the remaining attributes. The objective is thus to approximate the probability distributions of all attributes, given the observations on the actual values of some attributes, and prior probability distributions representing beliefs about the values of all attributes prior to observing any evidence. Rejection sampling extends the previously described forward sampling algorithm with a third step. In this third step, object diagrams containing attributes not conforming to the evidence are rejected. The sampling process ensures that root attributes always conform, but this is not the case for OCL-defined attributes.

Algorithm 2: Rejection sampling

```
1   for(int i=1; i<M; i++) {
2       x = sampleExistenceAttributes();
3       N = extractObjectModel(Op, x);
4       if (syntacticallyCorrect(N)) {
5           y = sampleRemainingAttributes();
6           Λ = assignAttributesToModel(y, N);
7           if (conformsToEvidence(Λ)) {
8               O.add(Λ);
9           }
10      }
11  }
```

As described above, rejection sampling extends forward sampling. In this way, it overcomes the weakness of only allowing the specification of evidence on the root attributes. The pseudo code above shows that this is implemented as a filter, where samples conforming to the evidence are kept and all others are rejected. This approach is costly, as it requires the creation of many samples to generate a sufficient number of valid samples.

Metropolis- Hastings sampling

Metropolis-Hastings sampling (cf. Algorithm 3) is an iterative sampling technique converging to a desired distribution limit. It aims at creating a Markov chain MC with a stationary distribution being the desired distribution, i.e., a chain of samples where the sampled attribute values match the specified evidence.

First, one valid sample is created using rejection sampling. Once this sample is found, it is used as the first element in the Markov chain.

Algorithm 3: Metropolis-Hastings sampling

```
1   Randomly create x_init
2   MC.add(Λ_init)
3   for(int i=1; i<M + B; i++) {
4       Λ′ = generateNewSample(Λ);
5       P(Λ′|Λ) = calculateProbability(Λ′, Λ);
6       α = calculateProbabilityOfAcceptance(Λ′, Λ, P(Λ′|Λ) );
7       if (α < l){
8           MC.add(Λ′)
9       }
10      else{
11          MC.add(Λ)
12      }
13  }
14  removeBurn-InSamples(B)
```

The second step is to create a new chain element based on the last added element. A new sample is created as a copy of the last chain element. For the attributes without any specified evidence new values are generated using a candidate-generating distribution. Then the likelihood of the new sample given the old sample $P(\mathbf{x}'|\mathbf{x})$ is evaluated. Thereafter the probability of acceptance α of the sample is calculated, considering the likelihood $P(\mathbf{x}'|\mathbf{x})$, which over time is given more weight to. If α is greater than a given limit l the sample is added to the chain otherwise the last added element is added again. The second step is repeated until a predefined number M of chain elements has been added. The first samples are typically not used to evaluate the model; they are called burn-in samples B and train the algorithm. As a final step the burn-in samples are removed.

Similar to rejection sampling, Metropolis-Hastings sampling allows specifying evidence for any attribute of the model. This algorithm does need a comparably smaller number of samples and is therefore more effective, especially when considering models including a large number of attributes. The biggest disadvantage of Metropolis-Hastings sampling is that, especially for models with many local minima, a solution not being the best one might be found. This is because of the chain structure of the result, where samples are based on their predecessor.

Table 5.6 compares the described sampling algorithms.

Sampling algorithm	Strengths	Weaknesses
Forward sampling	• Fast	• Only evidence on leaf nodes
Rejection sampling	• Evidence on all nodes	• Time-consuming for large modes
Metropolis-Hastings sampling	• Supports Evidence on all nodes • Relatively fast	• Risk of only finding local minima • Advanced configuration, requiring insights into the algorithm, needed

Table 5.6: Comparison of the sampling algorithms

5.5 Design option V: Level of abstraction

Many Enterprise Architecture models are large, consisting of several hundreds of entities (cf. Chapter 8). When Enterprise Architecture models are to be used for analysis purposes, they are typically even larger than models intended purely for documentation. This is the case because, typically, some extra information needs to be captured for the models as input for analysis. For example [182] presents a class diagram for the analysis of availability. This analysis theory introduces the usage of gates to aggregate available data between different layers of the architecture. These gates are not relevant from a documentation perspective but are needed for analysis.

As identified in the first section of this chapter, the provided level of abstraction needed to be selected under consideration of the following requirements of the tool:

- The tool shall offer a high degree of usability
- The tool shall possess presentation capabilities
- The tool shall support the creation of models

During the implementation of the tool and usage in practice, the high complexity of the models was identified. It was realized that to support the tool user in conducting an analysis, it is helpful to reduce the size of the graphical model to decrease the visual complexity and thereby facilitate the user's navigation through the model. To achieve this goal while retaining the previously selected modeling language UML and providing input to the P2AMF inference engine, three alternatives were identified.

First, the analysis theory described in the analysis framework could be simplified and shortened to reduce the number of entities of which the resulting models consisted. Second, the tool could leave the creation of easier comprehendible models to the user. Instead of an active support that would automatically be utilized during the tool usage, a passive approach triggered by the user was imagined. As soon as the user felt overwhelmed by the model, he or she could use a filter mechanism or rework the model, using cut and paste, to make it easier to grasp. The third alternative was support for templates. Using templates, model complexity can be encapsulated in building blocks, and the number of depicted model elements could be reduced. The same user defining the analysis theory could define the templates specifically per class diagram. This should lead to meaningful visual aggregations, as the author of the analysis theory likely has a good feeling for useful building blocks.

The three alternatives were evaluated, and both the second and third alternatives were implemented iteratively.

The first alternative, simplified analysis theory, would have contributed to a higher degree of usability. This approach would have supported the creation of models, and the resulting models would have consisted of fewer objects. The models based on the simplified theory would be easier to grasp, as their complexity would be lower. The tool would also have possessed improved presentation capabilities with this solution, as only a subset of the original model would have been visualized.

However, this approach also has some disadvantages that prevented its selection. The resulting models would have been simplified compared to models based on the original analysis theory. Therefore, analyzing the resulting models would at some point lead to different results from the models based on the original theory, as the used theory might not contain all relevant aspects. Another aspect is that the simplification of metamodels would have been difficult to realize using a generic approach. Instead, the simplification would be manual and time-consuming. This approach would also not solve the problem of complex models, only postponing it. Even based on simplified analysis theory expressed in metamodels consisting of fewer entities, it would be possible to create complex models; it would only take a longer time to do so. Finally, this approach would reduce presentation capabilities, as it would not be possible to present all information due to the simplifications.

In the first iteration, the previously described second alternative was realized. The tool offers the user the ability to define views [153] to visually slice and dice the model into smaller, more graspable sub-models. In addition, filters can be applied to depict only a subset of the contained attributes. It is also possible to only show the attributes that affect a particular attribute or only the attributes that are impacted by a certain attribute. A discussion of the tool functionality is, however, beyond the scope of this section. This is part of Chapter 8.

This approach contributes to the fulfillment of the requirement “the tool shall offer a high degree of usability”, as it is a flexible solution allowing the creation of individually tailored visualizations of subsets of the model. This approach also adds presentation capabilities to the tool, as relevant information can be presented to interested stakeholders based on filtering. The approach also supports the creation of models, as during the process of model creation, the focus can be set on a certain subset of the model. All other temporarily irrelevant parts can be hidden while modeling this subset.

The usage of viewpoints, views and filters has the disadvantage of adding another activity to the modeling process. The usage of this approach requires some manual activity and cannot be performed completely automatically. Someone, typically the theory expert, needs to define the relevant viewpoints, which might be time-consuming. Moreover, creating views illustrating subsets of the model also requires the investment of time on the part of the practitioner.

As a result of another iterative tool extension, the presented tool offers the usage of templates as a means of aggregation of model elements and thereby reduces the model complexity. This second extension, reducing the visual complexity, could be added to the tool without interfering with the previously described viewpoints and views. Instead, both features can be used jointly, increasing the usability even more.

The Enterprise Architecture analysis tool supports the specification of templates defined on top of the class diagrams. These templates can be used as building blocks during the instantiation of the class diagram into an object diagram. In particular, templates can be reused several times during the instantiation. In this way, it is possible to create a large object diagram fairly easy. This contributes to the

fulfillment of the requirement “the tool shall offer a high degree of usability”.

The creation of models is supported, as templates can be connected to other entities or other templates using defined ports (cf. Section 7.2). Instead of visualizing all constituent parts individually, the tool visually summarizes them into one model element and thereby decreases the number of shown entities. However, the tool user is still able to consider the elements that are included into a template individually, as the tool allows the consideration of the sub-model, which is described separately in a specific view. These features contribute to the presentation capabilities of the tool. Furthermore, the analysis results are not affected, as the tool internally dissolves the templates into its constituent objects before feeding the model into the inference engine.

The disadvantage of templates as they are supported in the tool at present is that their initial specification is time-consuming. The included classes need to be modeled before a template can be used, and it is also necessary to specify how the template can be connected to the rest of the model using its ports. The use of templates was first presented in [119].

As for filtering, the intent of this section is not to explain these features in detail. Instead, the design decisions and, in particular, the design decision to implement support for the tool-side reduction of the visual complexity are of importance. An explanation of the two mentioned features can be found in Section 7.2. As part of the discussion of the data model (cf. Section 7.3), the concepts of templates and viewpoints are related to the core information objects of the tool: the class diagram, representing the extended metamodel, and the object diagram, the instantiation of the metamodel.

Table 5.7 summarizes the comparison of the considered alternatives to support an appropriate level of abstraction.

Means for complexity reduction	Strength	Weakness
Simplified and shortened analysis theory	• Contributes to a high level of usability – Simplified models • Supports the creation of models – Models consist of less entities • Supports realizing presentation capabilities – Presentation of an aggregated subset	• Limited support the creation of models – Models are simplifications – No generic way to simplify models – Problem not solved only postponed • Limited support for realizing presentation capabilities – Not all relevant information might be available in simplified models

Continued on next page

Table 5.7 – continued from previous page

Means for complexity reduction	Strength	Weakness
Filtering combined with viewpoints and views	• Contributes to a high level of usability – Flexible solution allows individual tailoring of displayed model parts • Supports realizing presentation capabilities – Using of filters to provide visualizations for the target audiences • Supports the creation of models – Viewpoints and views can be used on top of existing models to focus on a subset	• Negative impact on usability – Either theory expert has to specify a viewpoint or practitioner has to manually create a view

Continued on next page

Table 5.7 – continued from previous page

Means for complexity reduction	Strength	Weakness
Templates	• Contributes to a high level of usability – Reuse of model components – Modeling of many objects at once • Supports realizing presentation capabilities – Level of abstraction is provided – Inside of templates can be considered if needed • Supports the creation of models – Templates behave like objects – Relation to other templates or objects	• Negative impact on usability – Specification time consuming – Needs to be done by a theory expert

Table 5.7: Comparison of means to provide a level of abstraction

5.6 Design option VI: Cyber security modeling

As mentioned in Chapter 1, the tool presented in this thesis was partly developed in the context of a research project with the goal of creating means for evaluating industrial control systems from a cyber security perspective. The project sponsors were particularly interested in developing tool functionalities to foster such analyses.

There are a variety of different approaches to security analysis [145], and no generic solution is applicable in every context. Therefore, an important decision, within the research project that the tool development was part of, was the selection of a cyber security analysis approach. The author was not directly involved in making this decision; however, the outcome was relevant for the presented work in that a means to support the chosen approach had to be designed and developed.

As stated in the introduction to this chapter, candidates for providing support for cyber security modeling needed to be evaluated regarding their contribution to the fulfillment of two requirements:

- The tool shall support the creation of metamodels covering the domains business architecture, information architecture, technology or technical architecture and solution architecture
- The tool shall support the creation of models

In accordance with the goal of the research project, the selected approach had to allow the making of enterprise-wide decisions. To produce input useable for the tool, the approach had to be expressible using the previously explained (cf. Section 5.3) modeling language of the tool (UML and OCL). It was also relevant that the selected approach was compatible with the method for Enterprise Architecture analysis (cf. Section 3.2), which is supported by the presented tool. This includes following a structure consisting of an extended metamodel specifying analysis theory and models instantiating this metamodel. Additionally, a consideration of uncertainty, as suggested by the analysis method, required. These latter requirements can be summarized as compatibility with the P2AMF (cf. Section 5.4).

Four families of approaches providing support for model-based security analysis while fitting into the context of the project were considered [247] techniques for model-driven security analysis during the system development, approaches to risk modeling, attack trees and attack graphs. The following comparison of the modeling approaches is a summary of the evaluation presented in [247].

SecureUML [158] provides a language with the purpose of specifying security aspects while performing model-driven development. This language comes with the drawback of lacking a method to quantitatively assess security based on the created models.

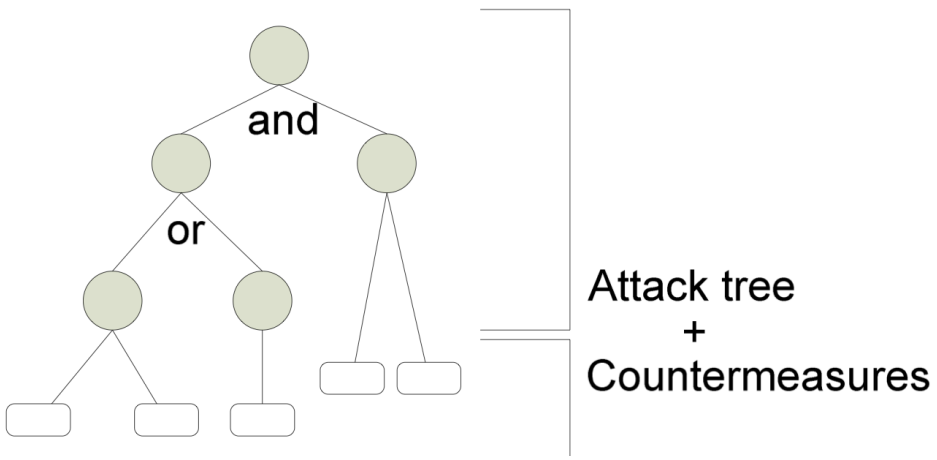
UMLsec [141] provides a language and methodology that can be used for security analyses during the development of systems. The output of the analysis method is a pass/fail result stating whether the architecture fulfills the requirements. Such a verification can support security risk analysis, but there are no automated means to compute security threats directly from the results [247].

CORAS [82] is a method for analyzing and quantifying risk. This is achieved by modeling the relationships between assets, threats, vulnerabilities, unwanted events, risks and treatments. The authors of [247] state that although risk in CORAS is defined as the product of likelihood and consequence, there is no analysis framework coupled to the metamodel and thus no algorithmic method to calculate risk based on a graphical description. Furthermore, there is also no description of which types of risk treatment should be modeled or how risk treatments influence risks in CORAS.

The concept of attack trees describes the application of fault tree analysis for cyber security analysis [230]. This approach aims at depicting the main goal of an attacker as the root of a tree. The steps that it takes to reach this objective are broken down into the subgoals of the attack. These subgoals can be related to each other using "AND" and "OR" gates. This is a frequently used way to model threats and security and exists in numerous variations [119, 246]. [233] suggests evaluating the probability of attacks taking place, the probability of success, whether special equipment is needed, the cost and the legality of mounting an attack on the system by assigning leaf events in the tree and then following their propagations upwards within the tree.

A holistic perspective on cyber security can be obtained when not only attacks but also countermeasures are considered [81]. [120] presents an approach in which countermeasures are modeled together with trees illustrating threats. [26] demonstrates the combination of countermeasures in a tree structure in so-called "defense trees". Figure 5.7 illustrates this approach.

Figure 5.7: Countermeasure Attack trees can be extended to Defense trees



Attack trees and defense trees have the disadvantage of being large. A fully comprehensive attack tree with all possibilities and factors would require one or

more experts and a large investment of time and effort, resulting in poor scalability [58].

In [247], it was described that attack graphs model the sequence of steps needed to accomplish the attack rather than the set of steps. This is modeled in a graph structure. [247] also reports that attack graphs have been used to assess the probability that an attacker reaches a particular attack step [131] or to analyze the security of system configurations in terms of the weakest adversary that can compromise the network [208].

[157] demonstrated the usage of Bayesian networks to express attack graphs to calculate the probability of an attack against computer networks being successful based on vulnerabilities within it. These "Bayesian attack graphs" can be used to answer questions about the current security status and to facilitate comparison with previous measurements [81]. Attack graphs have the disadvantage of possibly growing exponentially with the size of the network if no means to complexity reduction are used. This complexity makes the creation of visualizations difficult and has a negative impact on the usability of this approach [188].

Table 5.8 compares the discussed cyber security modeling approaches.

Security modeling approach	**Strength**	**Weakness**
Security evaluations during model-driven development (SecureUML, UMLSec)	• Supports the creation of models – Extensions to the UML	• Difficult to model business architecture, technology or technical architecture and solution architecture – Focus on the design of systems • No support for analysis

Continued on next page

Table 5.8 – continued from previous page

Security modeling approach	Strength	Weakness
Analyzing and quantifying risk	• Covers the domains business architecture, information architecture, technology or technical architecture and solution architecture – Relationships between assets, threats, vulnerabilities, unwanted events, risks and treatments	• Insufficient supports for the creation of models – No description of what different types of risk treatments that should be modeled, or how risk treatments influence risks • No algorithmic method to calculate risk based on a graphical description
Attack trees and defense trees	• Supports the creation of models – Goals and subgoals can be expressed • Covers the domains business architecture, information architecture, technology or technical architecture and solution architecture – Defense trees are a holistic approach to create enterprise-wide models	• Insufficient supports for the creation of models – Resulting models tend to be fairly large

Continued on next page

Table 5.8 – continued from previous page

Security modeling approach	Strength	Weakness
Attack graphs	• Supports the creation of models – Network structure between modeled elements can be captured • Covers the domains business architecture, information architecture, technology or technical architecture and solution architecture – Holistic approach to create enterprise-wide models	• Insufficient supports for the creation of models – Visualization can be difficult because of complexity – Attack graphs grow exponentially with the size of the network if no countermeasure is taken i.e. if no advanced approach is used

Table 5.8: Comparison of security modeling approaches

The other contributors to the cyber security research project of which the presented tool was an outcome chose to base their work on the usage of attack graphs. They initially created [247] a cyber security analysis framework making use of the PRM formalism. Later, as part of the research project for evaluating industrial control systems from a cyber security perspective, an updated and extended version was presented. P2CySeMoL: Predictive, Probabilistic Cyber Security Modeling Language [119] makes of the latest inference engine included in the tool (the Predictive, Probabilistic Architecture Modeling Framework (P2AMF)). While defining this language, the authors of [119] made use of the tools capabilities to address various levels of abstractions (cf. Section 5.5) to reduce the discussed complexity problems existing for attack graphs. They defined a set of templates, as introduced in the previous section on the level of abstraction, which could be used to describe typical reoccurring building blocks.

5.7 Summarized design decisions

The design decisions made are summarized in Table 5.9. For the overall architecture of the tool, implementing a common core and two specializations, each tailored to support one of the user groups, was chosen. The platform was realized based on the Eclipse Rich Client Platform. The Unified Modeling Language (UML) was selected as the modeling language. The inference engine of the tool was implemented based on the Predictive, Probabilistic Architecture Modeling Framework (P2AMF) [140]. To address the visual complexity arising from large Enterprise Architecture models, two support functionalities were added to the tool: the ability to filter and use of viewpoints and views to visualize tailored subsets of complex models and the ability to add templates. Templates, reusable submodels that are visualized as one model entity, allow the reduction of the amount of displayed content without minimizing the information used for analysis.

Finally, attack graphs were selected as a means to support the performance of cyber security analysis. This approach allows cyber security analysis on the network because the business architecture, information architecture, technology or technical architecture and solution architecture of an enterprises are described in one holistic model.

Area for making design decision	Outcome
Overall architecture	A common core and two specializations
Platform	Eclipse Rich Client Platform
Modeling language	UML
Inference engine	Predictive, Probabilistic Architecture Modeling Framework (P2AMF)
Level of abstraction	Filtering combined with viewpoints and views
	Templates
Cyber security modeling	Attack graphs

Table 5.9: Summarized design decisions

Chapter 6

Tool development process

This section describes the tool development process that was followed to implement the presented Enterprise Architecture analysis tool. The chapter concludes with a short presentation of significant milestones that were achieved during the tool development.

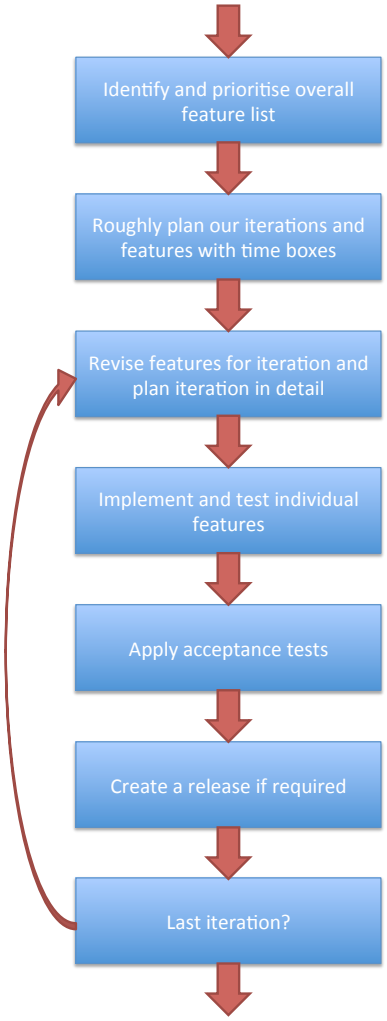
6.1 Development process

To ensure the successful implementation of the tool for the analysis of Enterprise Architecture models, the tool was developed closely with its intended users: academics specifying analysis frameworks and practitioners performing analyses based on analysis frameworks. An agile approach [1] was chosen to satisfy the users through the early and continuous delivery of valuable software [7]. Following an agile process model allowed the consideration of changing requirements, even late in the development [53]. It was thereby ensured that the tool always satisfied the stakeholders' latest requirements. Following this approach had the advantage of there always being working software available [53].

As the second group of tool users, i.e., practitioners that want to conduct analyses, depends on the first group, the theory experts, the latter was considered first. To perform analyses, an analysis framework needs to be in place, and in order for analysis frameworks to be available, a tool that allows the specification of analysis frameworks is needed.

This chain of impact led to a close collaboration between the tool development team and the theory experts during the initiation of the tool development. Once a minimalistic tool component for the specification of analysis frameworks was in place, the second user group, practitioners performing analyses, was considered as well. The tool component for the usage of analysis frameworks was then implemented, yet again in a basic version without advanced functionality. Once both tool components were in place, they were extended and modified iteratively in terms of

Figure 6.1: The overall structure of an FDD project as described in [121]



smaller and larger features. This approach was performed following a feature-driven development approach [121, 9] similar to the approach visualized in Figure 6.1.

The list of feature candidates contained potential additions identified by the author in collaboration with both analysis theory experts and practitioners. The author had the task of collecting potential new features, describing them and translating them into concrete work packages for the development team in the case of a selection.

At this stage, an evaluation of potential features was conducted as well. This

evaluation aimed at determining whether the feature requests were contradictory to the requirements of the tool presented in Chapter 4. If this was the case, alternative solutions were developed whenever possible. Feature requests carrying the risk of jeopardizing the goal of the presented research work were rejected.

The testing of new features was carried out in collaboration with the users who requested a certain feature to ensure that the implementation would meet the stakeholders' requirements. This was performed frequently, on a day-to-day basis depending on the feature, to ensure that no resources were wasted, in accordance with the Agile Manifesto [7]. Once a feature was implemented satisfactorily, a new release of the Enterprise Architecture analysis tool was released, and the feature was made available for all tool users. Once the implementation of a feature was complete, the technical documentation of the tool was updated accordingly, an activity not covered in Figure 6.1.

In accordance with [121], the feature prioritization was carried out in a larger group, led by the author. In this group, the development team, the stakeholders requesting new features and tool users without a particular need for a tool extension were all represented. Possible new features were evaluated with regards to their costs, scientific contribution, contribution for the practitioners and interdependencies. Whenever a feature implementation iteration was completed, a decision regarding the following tool extension was made.

Another activity that is not covered in Figure 6.1 is the aspect of bug fixing, especially concerning tool-wide bugs arising from the realization of a new feature. Sometimes the realization of a new feature had interdependencies with the existing features. It could be the case that old features stopped working as a result of changes made to realize new functionality. To handle situations such as this, a process parallel to the one described in Figure 6.1 was introduced. This process dealt with bug handling and, in particular, bug fixing. This process was always given highest priority so that newly identified bugs could be solved immediately, pausing the regular development process. This proceeding is in line with the Agile Manifesto and its goal of always having working software in place [7].

The development of the tool for Enterprise Architecture analysis became increasingly professional over time. Initially, it was a side project of a Ph.D. student. Later, it was conducted as a master thesis project, and then a full-time programmer started working on the tool. The project was supported by a number of students completing their internships or conducting their master thesis projects. Typically, the student projects had the task of developing proofs of concepts with a limited focus and functionality. The results of these test developments were then fed back into the main branch of the tool development.

To identify and realize new tool features, collaborations with both theory experts and practitioners were performed. Collaborative work with more than 10 frequent users applying the tool to specify analysis frameworks was performed over a period of six years. Additionally, collaborations with practitioners were carried out for over five years to capture their needs during the tool implementation. This was carried out in part through industrial case studies and master thesis projects. Following the

Agile Manifesto, these groups were also used to identify redundant and unnecessary functionality, which was then removed to ensure a simple and efficient software product.

Chapter 8 reports the application of the tool in collaboration with both previously mentioned audiences in detail.

6.2 Important milestones

This section briefly reports on some important intermediate tool versions that were implemented before or during the presented research project. The latest version of the tool is covered by this thesis; however, four predecessors are worth mentioning.

The first attempts to implement a tool for the analysis of Enterprise Architecture models were presented in [132]. As part of this research project, spreadsheets were used to analyze architecture descriptions.

In [135], the first independent tool for Enterprise Architecture analysis was sketched. The tool implemented the method for analysis of Enterprise Architecture models that was presented in Section 3.2. This tool featured the inference engine based on extended influence diagrams described in the previous chapter. The tool was already implemented in JAVA, a decision retained until the present study. The NetBeans Visual Library (cf. Section 5.2) was used.

In [64] and [44], the third version of the tool was presented, the result of the master thesis project described in [40]. This version used the same technologies as the previous version on the outside. However, it featured an inference engine based on probabilistic relational models (PRMs) [143]. The code base of this implementation was of significantly higher quality, and design patterns, including the Model-View-Controller pattern [217], were applied for the first time.

All three versions described above were prototypical implementations illustrating the general idea of tool support for Enterprise Architecture. However, these prototypes possessed limited usability and only basic functionality.

The fourth version of the tool was the first to provide sufficient stability and usability to allow large-scale usage. Some of the tool applications presented in (Chapter 8) were made with this version. This was achieved in part using an extended development team as well as a more structured development process that included the usage of test cases. For the first time, more than just basic tool functionality was implemented. For example, [42] described a tool feature that can be used to automatically instantiate class diagrams based on the results of vulnerability scanners. This tool was the first to support a P2AMF-based inference engine, which was presented in [259] (under its previous name pi-OCL). This P2AMF-based inference engine was offered in parallel to an inference engine based on PRMs. The filtering and tracing of attribute dependencies, as presented in the previous chapter (cf. Section 5.5), was offered as well.

The fifth and current version of the tool was first presented in [43]. Following the argumentation regarding the platform included in the previous chapter, this

version makes use of the Eclipse Rich Client Platform [167] and Eclipse Modeling Framework [250]. This version only offers the P2AMF-based inference engine that was developed for the previous version. Templates (cf. Section 5.5) were implemented as well. The fifth version of the tool has been used by more than 10 analysis experts and more than 50 practitioners (including students using the tools for the analysis of real and fictitious cases) to date.

For a more detailed description of the tool usage, the reader is referred to Chapter 8, which elaborates on the tool applications.

Chapter 7

Artifact

Following the used method (cf. Chapter 2), this chapter demonstrates the result of the research project described in this thesis from a technical perspective. This chapter starts with a description of the user interface of the Enterprise Architecture analysis tool, which was impacted by the design decision to implement this tool as two components sharing a common core. The Class modeler is the tool component that allows the specifying of analysis frameworks consisting of class diagrams representing analysis theory for Enterprise Architecture modeling. Further viewpoints and templates, both specified on top of the class diagram, can be included in the analysis framework as well. The counterpart of the Class modeler, the Object modeler, supports the instantiation of the class diagrams and contains the inference engine, which can be used to evaluate the architectural descriptions. For both components, a presentation of the suggested workflow, illustrating the recommended order for applying the functionality provided by the user interface, is presented as well. Thereafter, a presentation of the distinct functionality of the tool is given. The distinct functionality of the Class modeler is presented first. The description is followed by a presentation of the functionality of the Object modeler.

An in-depth description of the tool's architecture, internal structure and the used libraries is presented in the following. Finally, once the presentation of the tool is established, the contribution made by the author (cf. Section 1.2) is related to the various characteristics of the tool discussed.

The resulting artifact was developed based on the previously made design decisions (cf. Chapter 5). In Table 7.1, a mapping between the design decisions and some of the features of the tool is presented.

The decision to realize the overall architecture using a common core and two specializations was reflected in the design of the user interface (cf. Section 7.1). Furthermore, the used data model (cf. Section 7.3) was an outcome of this decision, as it was designed to connect the two components of the tool. The functionality to update the used analysis framework (cf. Section 7.3) is also a consequence of the architecture based on two components. This functionality is a connector between

the Class modeler and Object modeler and allows the analysis framework used to analyze architecture models to be refreshed.

The decision to use the Eclipse Rich Client Platform (cf. Section 5.2) impacted the architecture of the created artifact (cf. Section 7.3). This architecture was designed to optimize the utilization of the platform's capabilities.

The design decision to use an inference engine based on the Predictive, Probabilistic Architecture Modeling Framework led to the design of a functionality that allows validating input according to this framework (cf. Section 7.2).

The decision to support filtering combined with viewpoints and views and templates to handle the visual complexity of the models led to two features that are presented in this chapter: complexity reduction using templates (cf. Section 7.2) and the creation of filtered views based on viewpoints (cf. Section 7.2).

The decision to support cyber security modeling using attack graphs led to the development of the functionality to trace the source of impact (cf. Section 7.2) to find the reason for a given analysis result. This functionality can be used to identify security deficiencies and meaningful countermeasures while conducting cyber security analyses.

Area for making design decision	Outcome	Reflected in tool features
Overall architecture	• A common core and two specializations	• User interface • The used data model • Update of the used analysis framework
Platform	• Eclipse Rich Client Platform	• Architecture – The used platform – The resulting tool architecture
Modeling language	• UML	
Inference engine	• Predictive, Probabilistic Architecture Modeling Framework (P2AMF)	• P2AMF validation
Level of abstraction	• Filtering combined with viewpoints and views • Templates	• Complexity reduction using templates • Creation of filtered views based on viewpoints
Cyber security modeling	• Attack graphs	• Source of impact tracing

Table 7.1: Mapping between the design decisions and the reflecting tool features

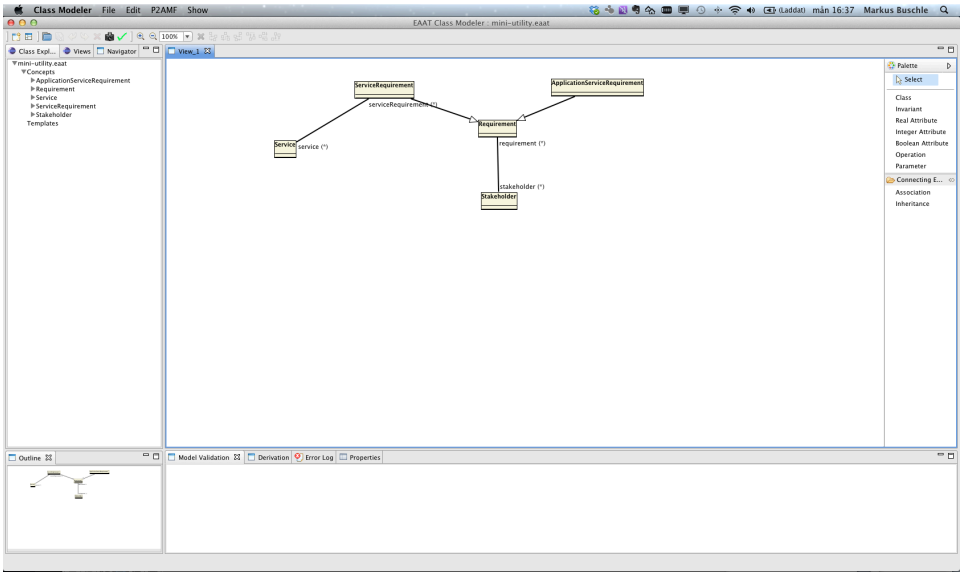
Not every feature of the tool that will be discussed in this chapter is directly determined by the design decisions. Some of the functionality that the developed tool for Enterprise Architecture analysis possesses was not mainly added to meet the previously identified requirements or to fulfill design decisions that arose based on the requirements. Instead, features were sometimes added as the need for them was identified during the application of the tool. However, the support of these features contributed indirectly to the fulfillment of the requirement "the tool should offer a high degree of usability" in that the added functionality either simplified the application of the tool or made the tool better at performing its tasks. Before new features were added to the tool, it was always ensured that they were not contradictory to the previously made design decisions.

7.1 User interface

This section contains a description of the user interface of the presented tool. Initially, a presentation of the user interface of the Class modeler is given. Thereafter, the typical workflow when using the Class modeler is presented. This is followed by a presentation of the Object modeler and the typical workflow for the application of this component.

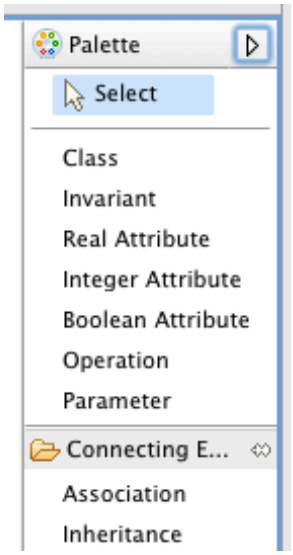
User interface of the Class modeler

Figure 7.1: The user interface of the Class modeler



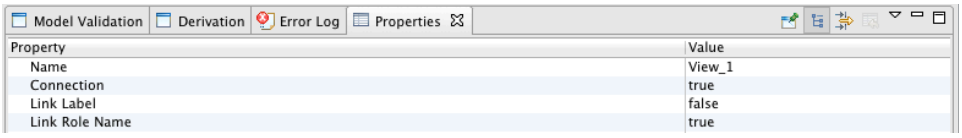
The user interface of the Class modeler (cf. Figure 7.1) mainly consists of the modeling canvas, which can be found in the center. This is the location in which, during the application of the Class modeler, the analysis framework and in particular the included class diagram will be created. To the right of the model canvas, a palette can be found (cf. Figure 7.2). This palette holds all of the possible model elements that can be used to construct an analysis framework.

Figure 7.2: The palette of the Class modeler



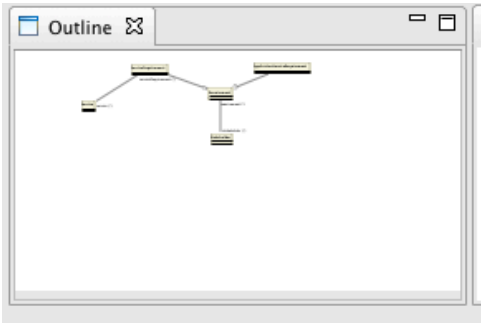
Below the modeling canvas is an area consisting of several tabs (cf. Figure 7.3).

Figure 7.3: Tabs to consider and specify model properties of the Class modeler



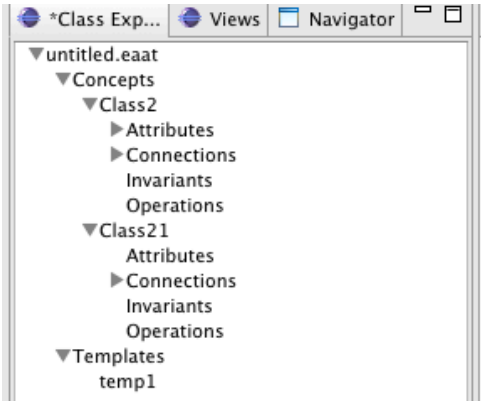
These tabs show the outcome of model validation, allow the specification of P2AMF-based derivations, visualize the error log and allow the investigation and changing of properties of the model elements. In the lower left corner, a model outline can be found (cf. Figure 7.4).

Figure 7.4: The model outline of the Class modeler



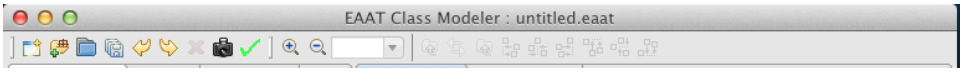
This outline contains allows the user to quickly navigate through the model and to consider it from a high level. To the left of the modeling canvas, three tabs can be found (cf. Figure 7.5).

Figure 7.5: The tabs allowing navigating through the Class diagram



These tabs allow the content of the created class diagram and its visual structure to be explored. One tab, the Class Explorer, lists all of the created classes, their attributes, relations and the defined templates. A second tab, Views, follows, allowing the consideration of different views on the model. The tab navigator allows the investigation of the currently considered view.

Figure 7.6: The menu bar of the Class modeler



On top of the modeling canvas, a menu bar can be found (cf. Figure 7.6) . This bar allows quick access to several frequently used functionalities. Both new Views and Templates can be defined based on classes of the model. New models can be opened, and the existing one can be saved (including Save As ...). Thereafter, the menu bar offers the functionality to undo and redo the last performed activity. Moreover, the menu bar offers delete functionality, export of the currently considered view/template to png and model validation. This functionality is followed by zooming functionality. Thereafter, the user interface offers several alignments options. The menu of the tool is located in the top area of the user interface. Here, the File menu offers to save, load and export the analysis frameworks. The Edit menu includes undo/redo, delete, cut, copy & paste and select all. In the P2AMF menu, a refactoring of the P2AMF code is offered. A switch to turn on autovalidation and an option to validate the model follow. Finally, the Show menu allows all of the dialogs and elements of the user interface to be shown again if they have been closed.

Workflow of the Class modeler

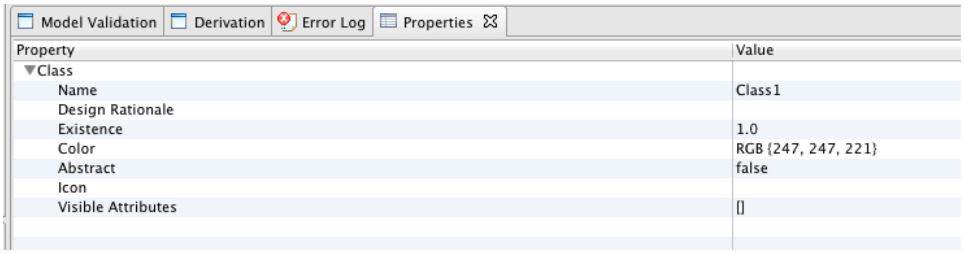
The creation of analysis frameworks typically follows the process described in Figure 7.7. A user can perform the process steps in any order or divided over several iterations.

Figure 7.7: The recommended process for the usage of the Class modeler

The following descriptions assume that the process described here is followed. The application of the Class modeler begins with the modeling of the relevant classes. Classes can be added to a model from the palette (cf. Figure 7.2). First, one needs to select a class and then drag it out to the modeling canvas. The name of the class can then be adjusted by double-clicking the top part of the newly added class. Additionally, it is possible to change the name from the properties tab (as described above, located in the lower part of the user interface). This tab (cf.

Figure 7.8), once a class was selected, also offers the option to specify a design rational, i.e., a short notice on the modeled class. The existence, according to P2AMF, can be specified, and the background color can be set. The class can be set to be abstract. It is also possible to assign an icon to that class. Finally, the visible attributes (for that particular view/templates) can be specified.

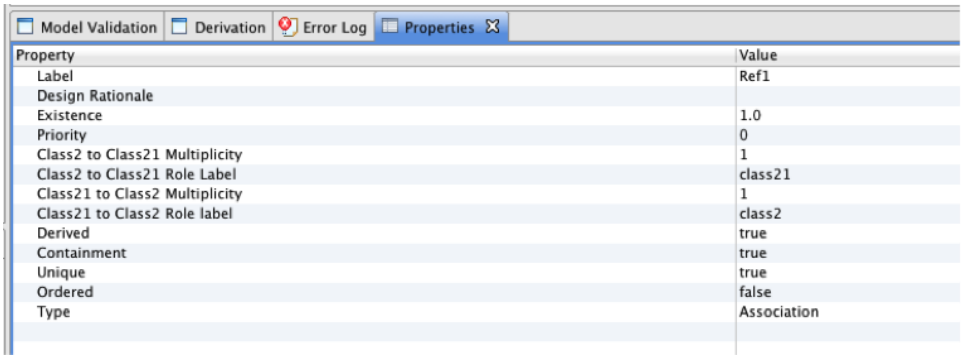
Figure 7.8: The properties tab for classes used during the application of the Class modeler



If a class should describe a specialization of an already modeled class, an inheritance relation can be used. From the palette, the connecting element inheritance can be selected for this purpose. This can, for example, be used to express that the class Linux is a specialization of the class Operating System. In a model, one would set an inheritance relation from Linux to Operating System.

Another way of connecting two classes is using the association relation. Associations can be created from the palette as well. Once Association is selected, relations will be created between any two classes that are selected in order. The properties of the association relationship can be modified from the properties tab (cf. Figure 7.9). The name of the association can be changed. It is also possible to specify a design rationale and to set the existence of the relation according to P2AMF. Likewise, multiplicity and roles can be set. Furthermore, OCL properties of the relation can be changed. The relation can be set to be derived, a containment, unique and ordered. Additionally it is also possible to switch from an association to an aggregation or composition relation. The derivation statement, in the case of a derived relation, can be specified in the derivation tab. It is possible to specify a priority for a derived relation. This priority specifies when the derivation will take place (compared to the derivation of other derived relations). The tool uses 0 as the highest value and all values below 0 (e.g., 1, 4, ...) as lower prioritizations. This can be helpful when relations should be derived considering other derived relations.

Figure 7.9: The properties tab for relations used during the application of the Class modeler



One can use attributes to describe classes in the context of the analysis framework. The Class modeler supports three different types: real, integer and Boolean. These three types can be found in the palette. Classes can be described using these attributes by dragging an attribute from the palette to the class of interest. Two types of attributes can be used: derived and non-derived. To specify how an attribute should be derived based on other attributes and/or the structure of the model, a derivation statement needs to be specified. This specification can be performed from the derivation tab. Attributes can be modified from their properties tab. Their name can be changed, a design rationale can be provided and whether the attribute should be derived and the type can be set.

To foster code reuse, P2AMF allows the definition of operations to be used during attribute derivation. To add operations to a class, one needs to drag the operation element from the palette to the object that should be equipped with it. From the properties tab, these operations can be customized. A new name can be set, and a design rationale can be provided. In addition, the output type can be specified (yet again, from real, integer and Boolean). It can also be specified whether the result should be a single element or a collection.

Parameters can be added to describe the input of operations. Yet again, this addition is performed from the palette. A new parameter can be dragged from the palette to the desired operation, where it then can be dropped. Once a parameter is dropped, it can be adapted based on the user's intention. From the properties tab, it is possible to change the name, assign a design rationale, change the type (real, integer and Boolean) and set whether the parameter should describe a set (including a bag and sequence) or a single element.

As described in Figure 7.7 it is possible to specify invariants on the model. Invariants are constraints on the model that need to hold during the creation of object diagrams. These can, for example, specify the values of attributes, number of instantiations or structure of relations. Yet again, these can be found in the

palette and can be assigned to elements of the model by dragging and dropping. Their evaluation can be specified in the derivation tab.

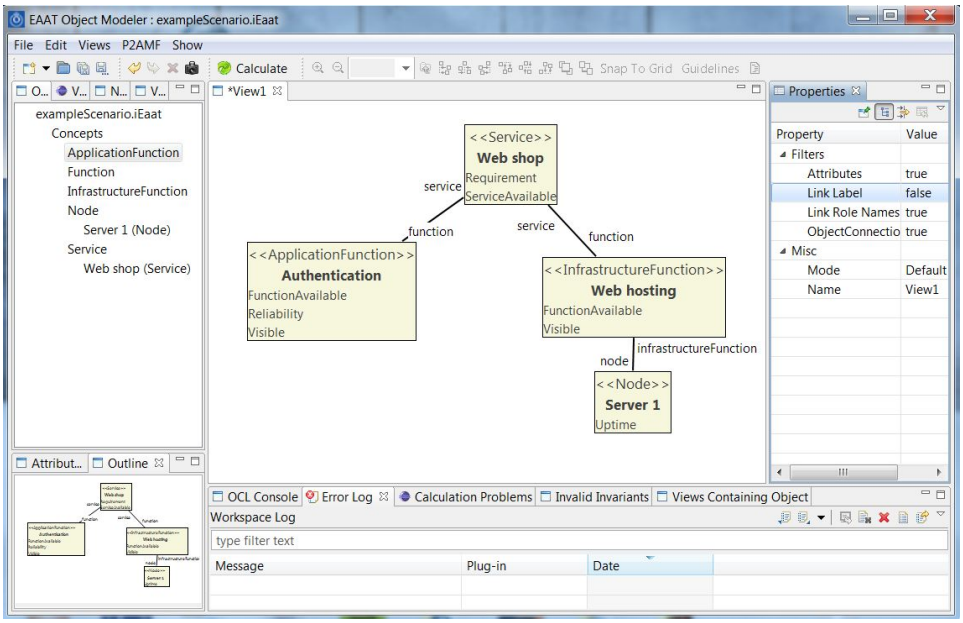
A click on the plain canvas allows general properties of the model to be set (cf. Figure 7.3). Again, this step can be performed from the properties tab. This tab allows the name of the currently considered view to be changed. Additionally, whether connections, link labels and role names should be shown can be set. The pattern to display role names can be varied as well. Additionally, the visibility of attributes, operations and invariants can be specified. Finally, it is possible to set this view as a viewpoint (to be used in the object modeler).

Once a model is complete or at any moment during the model creation, the user can validate it. Validation here refers to checking whether the model is a proper P2AMF model and can be used for calculations. This can be triggered from the P2AMF model. The results are displayed in the model validation tab, located in the lower part of the user interface. For validation, the Class modeler translates the model into an OCL-compatible model. Next, the Class modeler uses third party libraries (provided by the eclipse modeling framework EMF) to check the syntax and semantics.

User interface of the Object modeler

The design of the user interface of the Object modeler was based closely on that of the Class modeler. Therefore, yet again, the modeling canvas consumes the most space and is located in the center of the application. To the right of the model canvas, a palette can be found (cf. Figure 7.10) In the Object modeler, this palette visualizes the properties of the currently considered model element, regardless of whether it is an object, relationship or attribute.

Figure 7.10: The Object Modeler with the modeling canvas in the center

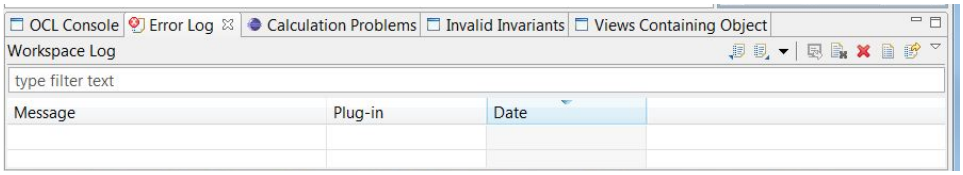


Below the modeling canvas, again, an area consisting of several tabs can be found (cf. Figure 7.11).

Figure 7.11: The menu bar of the Object modeler

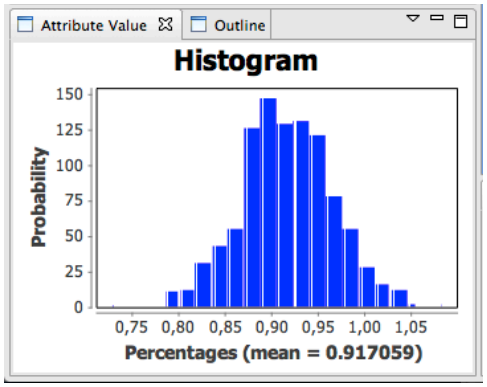
These tabs allow the manually querying of the created model from the OCL Console, reporting on errors during model creation and calculation. The user can identify invalid invariants and trace the usage of the currently considered object over the model's views. Yet again, as observed in the Class modeler, the lower left corner provides the user with a model outline (cf. Figure 7.12).

Figure 7.12: Tabs to consider and specify model properties of the Object modeler



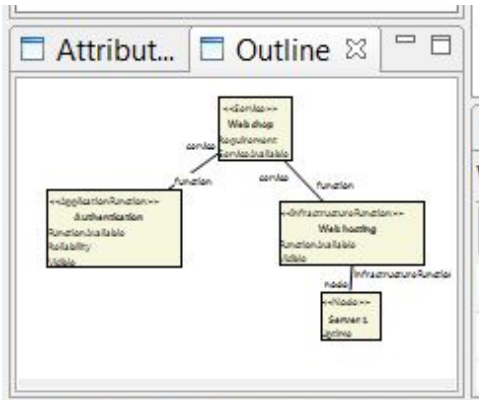
This outline allows navigation through the model and consideration of it from a high level. In contrast to the Class modeler, the lower left corner also contains an Attribute Value tab allowing the user to consider the value of a selected attribute (cf. Figure 7.13).

Figure 7.13: The Attribute value tab of the Object modeler



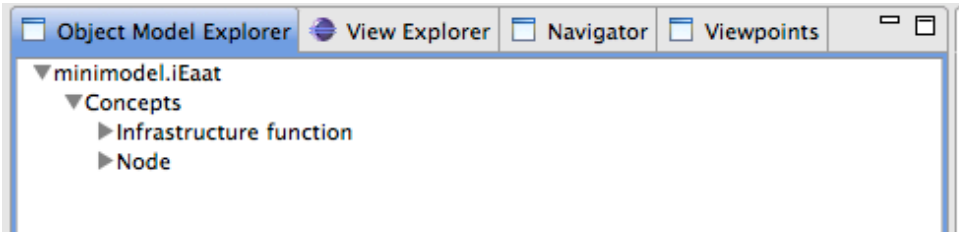
To the left of the modeling canvas, four tabs can be found (cf. Figure 7.14).

Figure 7.14: The model outline of the Object modeler



These tabs allow the exploration of the content of the created object diagram and its visual structure. One tab, the Object Model Explorer, lists all of the instantiated objects and their defining classes, their properties and the defined templates. A second tab, Views, follows, allowing the consideration of different views of the model. The tab navigator allows the currently considered view to be investigated. Finally, the tab Viewpoints allows the consideration of the viewpoints that were specified on top of the used class diagram.

Figure 7.15: The tabs allowing navigating through the Object diagram



Again, following the Class modeler, on top of the modeling canvas, a menu bar can be found (cf. 7.15) . New views can be added based on the class diagram, and new models can be opened and the existing one saved (including Save As...). Thereafter, the menu bar offers the functionality to undo or redo the last performed activity. Additionally, the menu bar offers delete functionality and export of the currently considered view to png. The analysis framework can be updated, and the created object diagram can be evaluated, triggered by the Calculate bottom. This functionality is followed by zooming functionality. Thereafter, the user interface offers several alignments options.

The menu of the tool is located in the top area of the user interface. Here, the File menu offers to save or load the Object diagrams. Moreover, the used analysis framework and its included class diagram can be replaced. It is also possible to export the model to Excel to post-process it after analysis. The class diagram that the current object diagram is based on can also be extracted. The Edit menu covers undo/redo, delete, cut, copy & paste and select all. The user can add new views from the Views menu. The P2AMF menu can be used to trigger the analysis of the current model. In this way, the tool will feed the currently created model into the inference engine and, once an evaluation is complete, display the values of the derived attributes. The user can steer the calculation from the P2AMF menu and, in particular, select the sampling algorithm that should be used. He or she can also deactivate the sampling completely. It is also possible to remove all evidence (cf. Figure 7.16) that has been collected for the modeled objects. It is also possible to turn on autovalidation of the described constraints on the model. Finally, the Show menu allows all of the dialogs and elements of the user interface to be shown again if they have been closed.

Workflow of the Object modeler

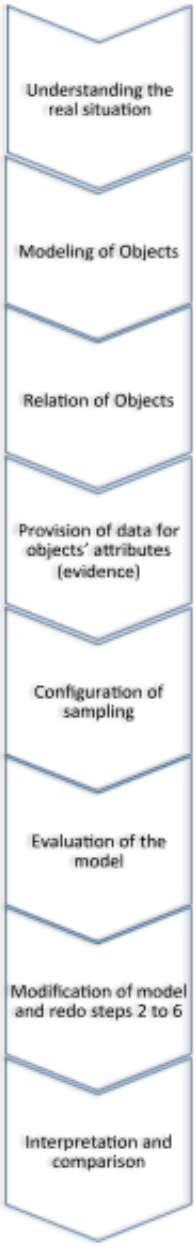
The first step in the usage of the Object modeler is to understand the situation that should be modeled in terms of the analysis framework that should be used. Thus, the user needs to perform a translation to determine how the real-life aspects that he or she wants to investigate can be modeled with the included class diagram. To

this end, the user needs to understand the analysis framework, the included class diagram and the scenario of interest.

Once the user achieves this understanding, he or she can start describing the scenario based on the class diagram included in the analysis framework. In this way, the user instantiates classes that are specified in the selected class diagrams into objects. Third, he or she connects the objects to describe the situation of interest considering the relationships specified in the class diagram. The fourth step is to replace the general data included in the class diagram with specific information for some or all attributes that are part of the model to provide a more specific description. In the nomenclature of probabilistic inference, such instance-specific data are called evidence [43]. Once the user is satisfied with the description of the situation (this does not necessary mean that the description is complete, for example, in the case of structural uncertainty), he or she can configure the sampling algorithm that should be used to infer the remaining, unknown attribute values (cf. Section 5.4). As soon as the sampling has been configured, the user can execute the sampling functionality of the tool, which will evaluate the model's attribute values autonomously. It is then the user's task to investigate the evaluated model and to consider its attributes. Based on the described scenario and in consideration of the performed analysis, the user can derive alternative setups of interest. He or she can represent them again as models following the described process. Finally, once the user has evaluated all relevant alternatives, he or she can compare his or her findings and identify one or several preferable scenarios.

The described process is illustrated in Figure 7.16.

Figure 7.16: The recommended process for the usage of the Object modeler



7.2 Distinct functionality

This section describes the selected features of the presented tool. They are discussed because they are unique, or at least rare, for Enterprise Architecture tools. Some of the functionality that the tool offers is inspired by solutions found in other tools; however, this tool extends these capabilities. Moreover, the presented features all support reaching the goal of this project, i.e., to create a tool for Enterprise Architecture analysis. The features of the tool that are common for almost all available software tools, such as the ability to save a model and load it at a later moment, are not part of this section. This category of features neither has scientific depth nor requires explanation upon usage.

In the remainder of this section, functionality of the Class modeler will be discussed first. Thereafter, the capabilities of the Object modeler worth mentioning will be discussed.

Distinct functionality of the Class modeler

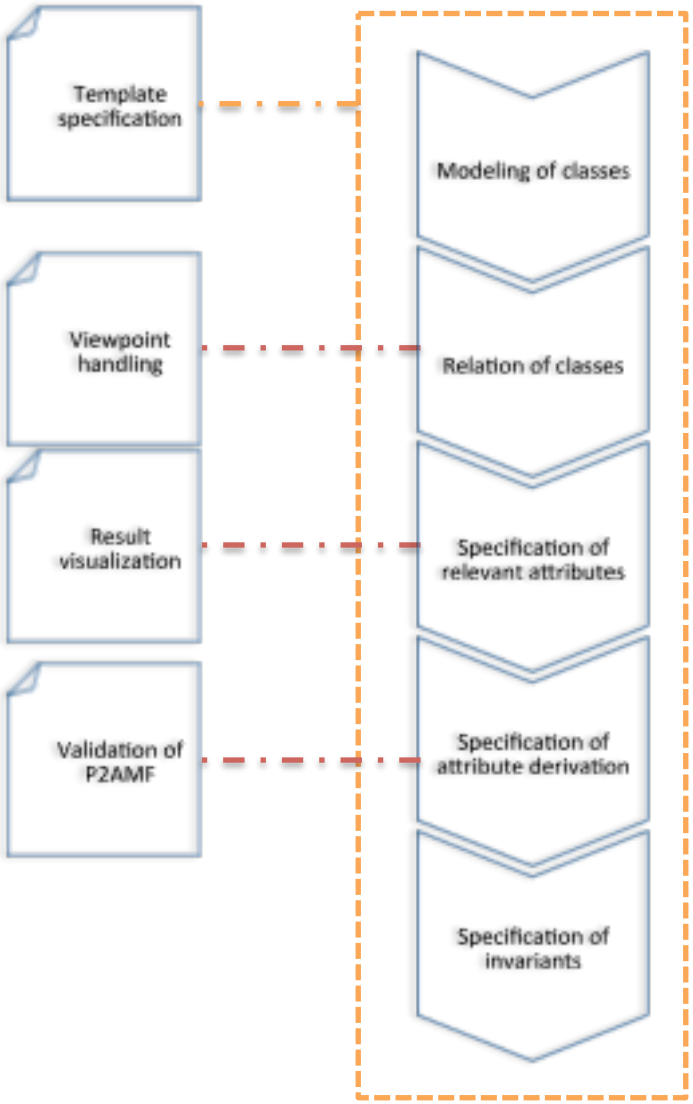
In this subsection, the functionality of the Class modeler distinguishing it from other Enterprise Architecture tools will be introduced and discussed. In particular, the handling of viewpoints, complexity reduction using templates, the configuration of result visualization based on the analysis outcome and the validation of the P2AMF code will be covered.

Figure 7.17 illustrates how the features that are discussed in the following relate to the workflow presented earlier (cf. Figure 7.7). The features for the specification of templates as well as viewpoints support the user of the Class modeler throughout the usage of this tool component. The feature to specify the result visualization on the class diagram level is especially relevant during the specification of the relevant attributes. Finally, the validation of the P2AMF code is useful when P2AMF is used to specify how attribute derivation should take place.

The features of the Class modeler that will be discussed in this section are:

- Viewpoint definition
- Complexity reduction using templates (definition)
- Result visualization
- Validation of P2AMF code

Figure 7.17: The workflow of the Class modeler and its relation to the presented features

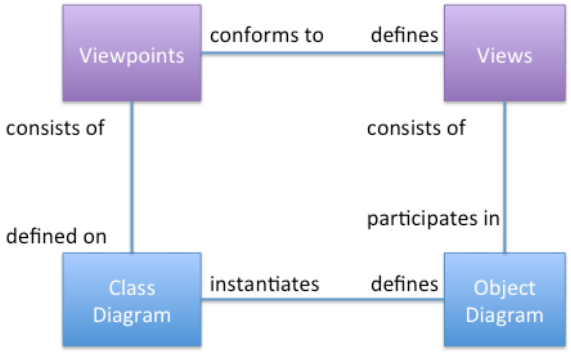


View point definition

According to the IEEE [116, 115], a viewpoint is a pattern for capturing a set of concerns in an architecture description. Views follow viewpoints and construct a

representation of a whole system from the perspective of a set of concerns. This is visualized in Figure 7.18.

Figure 7.18: The structure of viewpoints and views



Viewpoints are typically subsets of the class diagram and consist of some or all of its classes. They contain from zero to all relationships between the included classes. Common Enterprise Architecture tools have built-in viewpoints to describe the information, IT, application and business architectures [166]. These viewpoints depend on the used class diagram, included in a selected analysis framework, that the tool user selects. The tool vendors sometimes also implement viewpoints based on what they consider to be the users' needs. However, the vendor-specific viewpoints typically cover the four previously mentioned domains of information, IT, application and business architectures on various levels of granularity, sometimes using different wording [234].

The presented tool addresses the topic of viewpoints in a broader way. The usage of viewpoints is spread over the Class and Object modeler. The Class modeler allows the theory expert to define viewpoints that he or she considers to be relevant, and the Object modeler allows instantiating the class diagram conforming to the previously defined viewpoint. The unique feature of the Class modeler is that viewpoints do not necessarily cover information, IT, application and business architectures. Instead, they illustrate any subset of the class diagram that the creator of the analysis framework that includes the class diagram considers to be worth investigating.

These can be subsets of classic viewpoints, e.g., only network links and routers of the IT architecture, as well as more unexpected combinations, such as some business aspects combined with some parts of the infrastructure architecture. Filtering on the attribute level can be performed. The user of the Class modeler can, for the classes that are part of a specified viewpoint, decide to only select some of the classes' attributes and some relationships.

The motivation for offering the advanced viewpoint feature is to support analysis, which is in line with the purpose of the tool. In particular, the intent is to

increase the ease with which the user can conduct analyses by reducing the visual complexity and to only display the information needed for a certain purpose (cf. Section 5.5). The values of attributes in Enterprise Architecture models often depend on the values of other attributes (cf. Section 3.2). These dependency chains are not necessarily limited to a certain class or a certain part of the architecture. Instead, complex network chains between the attributes can be found, for example, in the cases discussed in [260] and [119]. Using the Class modeler, one can reflect these chains as viewpoints and thereby help the user of the resulting class diagrams to trace chains of dependencies and conduct analyses. This feature is of particular interest for the creation of large and complex class diagrams, such as [136], or class diagrams that cover tradeoff aspects between several system properties of the model, such as [51] and [181] . In the case of tradeoffs in particular, it is helpful to investigate the considered properties in isolation, using viewpoints, before finally looking at the focal point.

Complexity reduction using templates (definition)

Another tool feature that helps reduce the visual complexity is the use of templates. This feature also accelerates the modeling process, as large and repetitive parts of the model can be described rapidly and automatically.

During an Enterprise Architecture modeling endeavor, the same submodel often needs to be created repeatedly. For example, a submodel consisting of a desktop computer with an operating system and office suite installed, an assigned role operating the machine and a connection to the enterprise network can be found several times as part of a company-wide model. The presented tool allows such a submodel to be described as a template for reuse. Templates are specified in the Class modeler as part of the specification of the analysis framework. Instead of adding and relating the involved model objects manually, the user of the object diagram can simply instantiate a template that is part of the analysis framework, and the tool will add all included components and their relations to the model. The tool visually aggregates the elements of a template into one component, resulting in a lower number of displayed objects compared to a manual modeling. However, these aggregates can be opened to more closely consider the included elements or to provide specific evidence for included attributes. Once the user has completed his or her detailed consideration, he or she can close the template again. To ease the analysis, the Class modeler also allows the specification of representative attributes. These attributes can be selected from the set of attributes that are part of the template and will be visualized as though the template were a regular class and the representative attributes were the attributes of that class. Using representative attributes, the user of the Object modeler can evaluate relevant aspects of the model without having to open up the template.

The tool allows defining ports similar to the ports suggested by SysML [83] to specify how the elements of templates interact with their environment. This is of particular relevance when an object should be connected to a template as part

of the instantiation of the class diagram. Using ports, it can be described that external objects should be able to connect to a certain object inside the template, all objects instantiating a certain class or a subset.

The user of the Class modeler can also specify default values for the attributes that are part of the created models. Furthermore, he or she can lock templates to prevent the user of the Object modeler from performing modifications.

Finally, the tool's architecture to support templates is also designed such that templates can be parameterized during the instantiation of templates. In the example of the desktop computer that was described earlier in this subsection, a parameterized instantiation allows describing a template with n roles assigned to the computer. Upon instantiation of the template, the Object modeler can then provide the correct number of roles given the specific submodel. The resulting template instantiations will then connect the appropriate number of role objects to the desktop computer object. This feature can also be used to describe generic workspace environments where computers are running different operating systems. Based on the provided operating system parameter, the tool is able to instantiate the corresponding service objects that the operating systems typically provide.

Result visualization

Enterprise architecture models tend to consist of numerous objects, each having a number of attributes. The analysis of such models by looking at one attribute at a time is a time-consuming and repetitive task. Instead, it is preferable to consider some or even all of the included attributes at a time. To obtain a quick overview, the absolute values of the Enterprise Architecture are often of minor importance, whereas a tendency to see whether the attribute values are within the bounds is helpful.

The Class modeler lets the creator of the class diagram specify value ranges for the included attributes. These value ranges can be assigned with colors. The Object modeler, after evaluating the model to derive unknown attribute values, uses this color mapping to visualize the state of the attributes. Doing so, the user of the Object modeler can easily grasp the state of the currently considered scenario. This feature can be combined with the usage of representative attributes, as described earlier.

P2AMF validation

Some class diagrams included in analysis frameworks use a large amount of P2AMF code to specify the performance of system property analysis. For example, [119] presents a class diagram for cyber security analysis consisting of more than 5210 lines of code. Dealing with such a huge code base can be cumbersome and error-prone, especially as the presented tool requires specifying code on an attribute or operation level (cf. Section 7.1 and 5.3). Therefore, it is difficult to ensure a high code quality with consistent wording and free of spelling mistakes or typing errors.

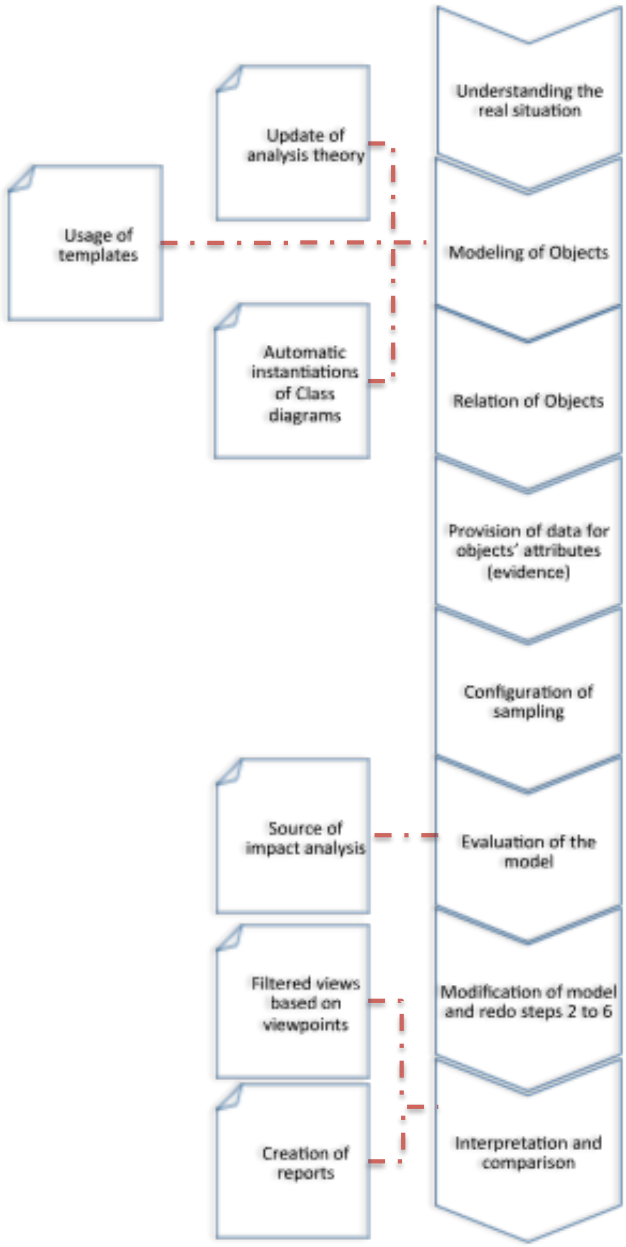
Validation is a well-established technique for ensuring the quality of a product within the overall software development lifecycle [65]. In the tool, validation is used to ensure that the P2AMF statements provided by the creator of the analysis framework are correct. To this end, the tool offers a validation feature inline with [190] stating that a validation assessment aims to determine the degree to which a model is an accurate representation of the real world from the perspective of the intended uses of the model. Applied to P2AMF, this means that the tool offers a functionality to test whether the provided P2AMF code is correct. As expressed in Section 5.4, P2AMF is built on top of OCL. Therefore, the tool translates the P2AMF code into OCL and tries to execute it using the Eclipse Modeling Framework (cf. Section 7.3). The tool then notifies the user regarding the outcome of the execution and points to existing code segments that contain errors.

Distinct functionality of the Object modeler

The previous section contained a discussion of the prominent features of the Class modeler component of the tool. This section covers the functionality of the Object modeler that is unique or at least exceptional in the way it is presented. This section addresses the usage of views based on viewpoints to filter the displayed part of the model, complexity reduction using templates, analysis to identify the source of impact for a certain attribute value, updates of the analysis framework, automatic instantiations of the used class diagram and the creation of reports to export calculation results.

Similar to Figure 7.17, Figure 7.19 illustrates how the tool features that will be presented in the following relate to the workflow of the Object modeler (cf. Figure 7.16). The feature supporting the update of analysis frameworks contributes to the modeling of object activities, as objects will be described according to the latest analysis framework. The feature usage of templates also contributes to the modeling of objects, as it accelerates the modeling process and generates a homogeneous structure. Third, the feature automatic instantiation of class diagrams contributes to the modeling of objects as well and reduces the time spent on modeling creation. It also improves the quality of the created model. The feature source of impact analysis contributes to the step evaluation of the model, as it allows the cause of a certain result to be traced. The filtering of views based on viewpoints contributes to the interpretation and comparison activity, as it allows the model to be looked at from different angles and with focus on various areas. Finally, the feature creation of reports contributes to the interpretation and comparison activity as well as it allows the analysis result to be exported and made accessible to interested audiences.

Figure 7.19: The workflow of the Object modeler and its relation to the presented features



The following features of the Object modeler will be discussed in this section:

- Creation of filtered views based on viewpoints
- Complexity reduction using templates (application)
- Source of impact tracing
- Update of the used analysis framework
- Automatic instantiation of the Class diagram
- Creation of reports

Creation of filtered views based on viewpoints

Large Enterprise Architecture models with several hundred objects are often difficult to grasp. Considering them as a whole is almost impossible, as there are too many model components and relationships between them that need to be considered. Looking at subsets of the model, for example, just the part of the model that affects a certain property, is often the only way to cope with this complexity. Following the architecture presented in Figure 7.18, viewpoints can be used in the presentation to visualize such subsets of interest. In particular, views can be created at any time during the usage of the Object modeler. To create a visualization of the submodel of interest, the user can create an empty view based on a defined viewpoint and copy and paste the complete model into this view. The tool will then work as a filter and test all of the elements that are pasted into the view. Only those model elements that are in accordance with the underlying viewpoint will be added to the view; the other objects will be rejected.

Complexity reduction using templates (application)

The templates included in the analysis framework and defined on top of the class diagram (cf. Section 7.2) can be used in the Object modeler to accelerate the modeling process. The user can select them as though they were regular classes and add them to the model. An instantiation of a template results in an instantiation of all of the classes that are part of it. As described previously, templates can be parameterized, and if this is the case, the tool first prompts the user to provide input for the parameters. Thereafter, these parameters are used to create a corresponding template instantiation.

Connecting templates to other elements of the object diagram, including to other templates, is realized under consideration of the relations that are defined in the class diagram. Only defined relations can be used. Moreover, this set is even more limited, as templates use ports to connect. Relationships that are allowed by the ports of the templates can actually be connected. The usage of templates is illustrated in the following:

Figure 7.20: A minimalistic Class diagram

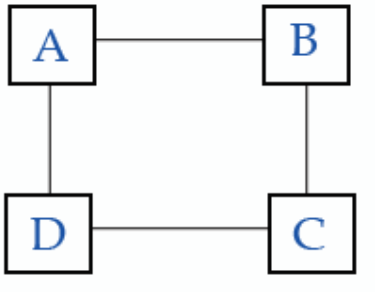
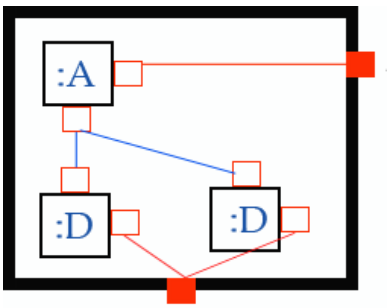


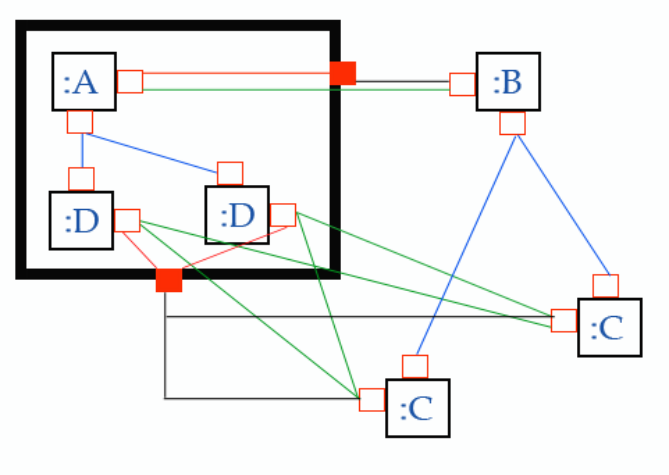
Figure 7.20 presents a minimalistic class diagram. A potential templates defined on top of this template could include that illustrated in Figure 7.21. This template contains one instance of the A class specified in the class diagram and two instances of the D concept.

Figure 7.21: A possible template based on 7.20



The template has two ports to connect with the environment (depicted as red squares in the border of the template). Following the class diagram, the user of the object diagram could either connect objects instantiating B (using the upper port) or objects instantiating C (using the lower port) to the template. Depending on the configuration of the ports (and the class diagram), several or just one B can be connected to the template. For the Cs, the port not only determines the number of objects that can be connected to the template but also to which objects the connections can be established. It could be that every C is automatically connected to all Ds or one specific D or that the user must make this decision manually. Using the ports to connect objects and templates leads to indirect connections between the objects and the objects that are included in the template. These indirect connections are colored in green in Figure 7.22, illustrating the described scenario.

Figure 7.22: An Object diagram illustrating the combined usage of templates and objects



During the usage of the Object modeler, the user can open the templates to investigate the constituent parts. If the template is not locked by the creator of the class diagram, the user can override the default values for the included attributes. Once the calculation functionality of the tool is executed, the tool resolves the templates into their constituent objects. Thereafter, the actual sampling process starts.

Source of impact tracing

In the presented tool, the focus is on the analysis of architecture models. The tool allows the inference of unknown attribute values based on information available on the states of other attributes. For a decision-maker using Enterprise Architecture as support in his or her work, it is important to know in which state a certain attribute is and to identify the reason for the outcome of an analysis. In particular, it is useful to identify the attributes of the model that actually affect the attribute of interest. Changing the values of these parental attributes might improve the situation, and it might be worth considering whether a scenario leading to higher values is worth adopting.

The tool allows the sources that lead to a certain result to be traced. For every attribute that is part of the model, a network over the attributes impacting it can be displayed. In this way, the determining factors can be identified. To create such networks, the tool uses the P2AMF code specifying the derivation of the attribute values. Following a bottom-up approach, i.e., going from the attribute of interest to its parents and then continuing to their parents, the P2AMF code is internally

unrolled and traced recursively until all attributes that are not dependent on the values of other attributes have been identified. The result is then presented to the tool user in a separate view only consisting of the network of impacting attributes. This gives the user the opportunity to experiment with the involved attributes and to identify one or several to-be scenarios that best fit his or her requirements.

Update of the used analysis framework

The creation of Enterprise Architecture is a time-consuming and expensive task [85], as many roles and many information sources need to be considered to create a holistic and complete description. Companies try to avoid performing this task frequently, instead aiming for the reuse of created architecture models.

As stated in (Section 3.2), there is often a certain degree of uncertainty involved in the usage of analysis frameworks used to perform Enterprise Architecture analysis. This uncertainty encourages the creator of a certain class diagram describing an analysis framework to improve this diagram based on his or her findings and insights gained during the application of the class diagram. Users of the Object modeler are typically prone to using the latest version of a certain class diagram, as it reflects the latest state of knowledge and has the lowest theoretical uncertainty. However, as described previously, creating a new object diagram each time the underlying class diagram is updated is not an option, as it is too expensive.

The tool overcomes this weakness by allowing the tool user to automatically update the analysis framework, i.e., the included class diagram. A user of the Object modeler can, at any time, refresh the class diagram and update the included P2AMF derivation rules. The tool does this without any need for manual interaction with the user. Internally, the Object modeler keeps track of the objects and the classes that they are instantiating. When an updated class diagram is loaded, the tool uses its database and compares for each class the latest available version with the one currently in use. If the Object modeler identifies a deviation, it alters the instantiating objects and performs an update of the P2AMF code.

Using this feature, there is no need for the user of the Class modeler to recreate his or her model; instead, the Enterprise Architecture can continue from the last version of the architecture model.

Automatic instantiation of the Class diagram

As stated in the previous subsection the creation of Enterprise Architecture models is time consuming and expensive. It is also error prone and consists typically of a lot for manual activities [70, 42]. Reducing the manual tasks is therefore preferable to cut costs and increase the data quality.

The presented task addresses this desire by a mechanism that supports the automatic creation of Enterprise Architecture models. In particular, it allows the instantiation of the used class diagram automatically provided that a (partial) architecture description is available. [42] illustrates the automatic creation of cyber

security models and [118] for general Enterprise Architecture models, this is possible when, for example, the output of a vulnerability scanner is available. These tools probe networks for connected computer systems and the services they expose to the components attached to the network. The results can often be exported in xml format, following a (XSD) schema definition [84].

The tool allows the definition of mappings between those schema definitions and the class diagrams that should be used. Using this mapping, the Object modeler component can then instantiate classes and relate the resulting objects to each other, following the structure of the xml file.

As reported in [118], this approach has advantages over manual model creation in regard to both the data accuracy and the time spent creating architecture models.

Creation of reports

To ease the communication of analysis results and as a means of documentation, the Object modeler offers to export the outcome of Enterprise Architecture analyses as PDF files.

For the creation of reports, the tool iterates over all modeled objects, queries them for the values of their attributes and creates visualizations of these values. Finally, the visualizations are added to a PDF file, which is saved at a user-defined location.

7.3 Architecture of the tool

This section presents the (software) architecture of the presented tool. The section starts with a discussion of the used rich client platform. Thereafter, its application and the resulting tool architecture are presented. This section also contains a description of the data model describing how information is presented internally. As the resulting architecture includes the usage of several external libraries and components, a description of the utilized third-party components is presented in the following.

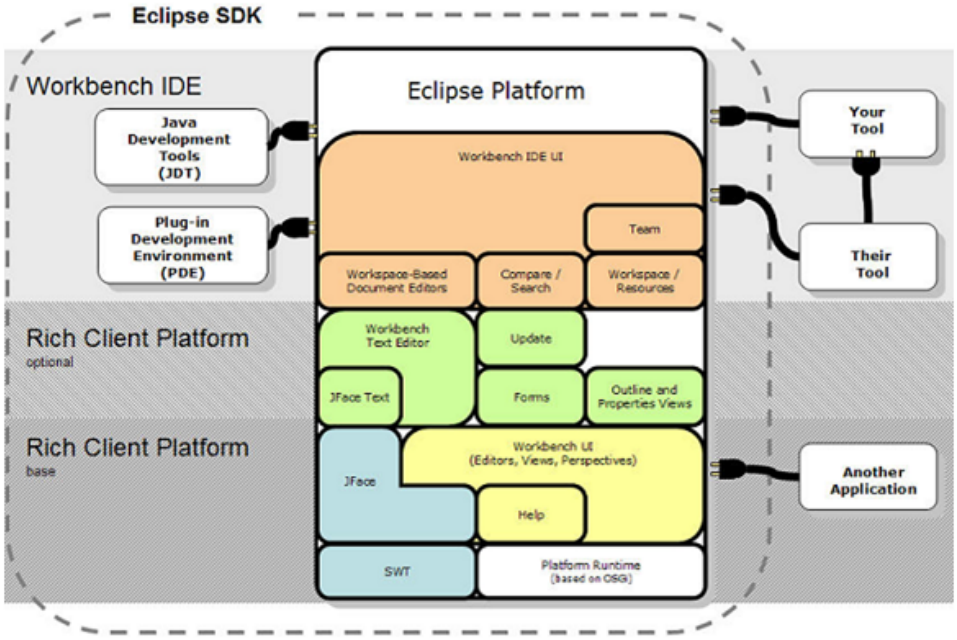
Used platform

Eclipse is an open-source community most known for its Java Integrated Development Environment (IDE) of the same name [167]. Underneath this IDE is a generic tooling platform supporting a variety of tools for languages and systems. Another level below is the Eclipse Rich Client Platform (RCP), a generic environment for executing applications. The previously mentioned IDE is one such application. Figure 7.23 illustrates the architecture of Eclipse.

This platform was used for the realization of the presented Enterprise Architecture tool. Eclipse has several features that made it particular interesting.

Components: During the implementation of the tool, the RCP component model [167] was heavily utilized. This model allows large applications to be com-

Figure 7.23: The Eclipse Rich Client Platform [31]



posed based on smaller components called plug-ins. Plug-ins can be reused, and several plugin-versions can be installed in parallel, a feature that supports the evolution of applications over time in particular[167]. The Eclipse RCP provides the user with a middleware layer simplifying the communication between plugins, as this does not need to be implemented per application.

Native user interface: The RCP integrates directly with the native user interface [167], resulting in applications that look and feel like other software programs for the same operating system.

Portability: Applications based on Eclipse RCP can be executed in any environment and operating system that offer a Java Virtual Machine. Currently, this applies to more than 95 % of all enterprise computers [199].

Stand-alone applications: Compared to thin client platforms, Eclipse aims at fat clients that are useable without a network connection. An RCP-based application contains all of the necessary libraries and components so that it can be used right away.

Component libraries and community: The Eclipse RCP comes with numerous plugins that the user can integrate into his or her own applications. These plugins include Help content, installer and update support, graphical and textual editing frameworks and reporting. An active community that works in close contact with the users develops these plugins. Therefore, found bugs can often be

acknowledged and solved within a short period of time.

Resulting tool architecture

Figure 7.24 describes the architecture that was derived. It contains three layers, in which the lower two (Rich Client Platform and Workbench IDE) originate from the Eclipse Rich Client Platform. The top layer, the EAAT (Enterprise Architecture analysis tool) IDE, represents the components implemented to support the performance of Enterprise Architecture analysis. The components that are used from each layer, as well as the own developed components that can be found in the EAAT IDE layer, will be discussed in the remainder of this subsection. The description will be given following the bottom-up approach because components of a higher level typically make use of the underlying functionality.

On the lowest level of the Eclipse Rich Client Platform, the Standard Widget Toolkit (SWT) can be found. SWT is an open-source widget toolkit for Java, which connects applications to the user-interface facilities of the operating systems.

On the same level, one can also find the platform runtime environment, OSGI - Equinox. This is the Eclipse implementation of the OSGI specification [8] and provides applications with a platform for network-provisioned components and services. It also contains a security model and features dynamic updates of the components built on top of it with minimal perturbation of the running environment [168, 105].

JFace is a toolkit used to create user interfaces. The toolkit is built based on SWT. It offers common components of user interfaces, such as dialogs, buttons and text fields [110]. The Eclipse Rich Client Platform also offers the Workbench UI [262], which allows the creation of workspaces in desktop applications. These workspaces can be seamlessly integrated in the application. Workspaces based on the Workbench UI contain one or several perspectives that control views and editors and define the menu structure. Applications can hold several workspaces at a time [39].

On the highest level of the Eclipse Rich Client Platform, one finds JFace Text, a framework based on JFace supporting the manipulation of text documents. JFace Text is used in the text editors provided by the Work Bench UI. Forms [242] is a framework complementing JFace and SWT for creating portable web-style user interfaces across the Eclipse user interface. Two specialized views, Outline and Properties views, are offered by the Workbench UI. The Outline view is used to show the objects presented in the editor, while the Properties view is used to edit the properties of the selected object. Finally, Update allows the parts of the implemented tool to be updated without affecting the other components.

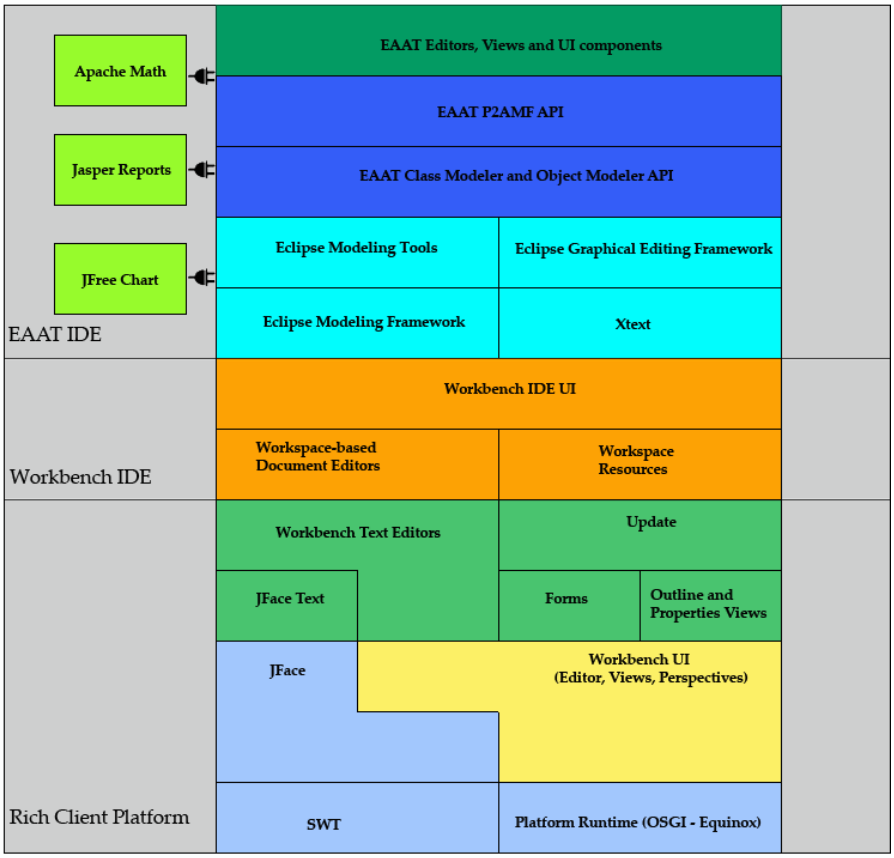
The intermediate layer of the developed architecture is the Workbench IDE. Here, the Workspace-based Document Editors can be found. These editors are specialized editors, built on top of the previously discussed Workbench Text Editors. The Workspace-based Document Editors are connected to files residing in the workspace. Any change in the editor will be recorded, and upon saving, the connected file will be updated accordingly. The Workbench IDE also contains the

Workspace Resources. These describe the information concerning the presented tool object and class diagrams that the user works with. The Workbench IDE UI is the user interface of the tool that integrates the Workspace-based document editors, resources and other user interface functionality provided by the Eclipse Rich Client Platform layer.

The highest layer in the architecture of the tool is labeled as EAAT (Enterprise Architecture analysis tool) IDE in Figure 7.24. To implement the tool, the Eclipse Modeling Framework (EMF) was used (cf. Section 7.3). EMF provides the tool with the ability to create ECore-based models and metamodels. In the tool, EMF was used to create class diagrams and their instantiating object diagrams. The Eclipse Modeling Tools, which can be used on top of the EMF, provide the tool with an OCL environment that was used during the implementation of P2AMF.

Eclipse Xtext is a framework that can be used to define own programming languages within an Eclipse environment. In the tool, Xtext was used to extend the OCL capabilities of the Eclipse Modeling Tools to feature probabilistic distributions as they are frequently used in P2AMF (cf. Section 5.4). Additionally, Xtext was used to realize other operations that are not part of the OCL standard. For example, operations to solve polynomials, needed for utility analyses as presented in [204], were added. To realize the user interface, widgets provided by the Eclipse Graphical Editing Framework [227] graphical editors and views for the Eclipse Workbench UI were also used. The Eclipse Graphical Editing Framework is an API that allows the creation of interactive diagrams and the visualization of information. The tool also uses JFreeChart (cf. Section 7.3) for the result visualization in terms of histograms and other diagrams. JFreeChart displays the results obtained from P2AMF evaluations performed using the Eclipse Modeling Tools. On top of the Eclipse Modeling Tools and the Eclipse Graphical Editing Framework, the Enterprise Architecture analysis tool Class modeler and Object modeler API can be found. These two APIs were designed with the goal of offering mechanisms for the creation of class diagrams and their instantiation in terms of object diagrams. On top of the Enterprise Architecture analysis tool Class modeler and Object modeler API, the Enterprise Architecture analysis tool P2AMF API was built. This API realizes the inference engine. It connects class diagrams and object diagrams to the tool component that implements sampling according to P2AMF, as described earlier (cf. Section 5.4). Furthermore, the Enterprise Architecture analysis tool P2AMF API makes use of the ApacheMath library (cf. Section 7.3) to realize the probabilistic functions that are fundamental parts of P2AMF. The library also connects to JasperReports (cf. Section 7.3) to export the outcome of the sampling in terms of reports. The topmost level of the EAAT IDE architecture layer consists of Enterprise Architecture analysis tool Editors, Views and UI Components. These components form the user interface of the tool. They allow the user to perform the previously described workflows (cf. Section 7.1) and provide the user with the functionality described in Section 7.2.

Figure 7.24: The resulting architecture



Data model

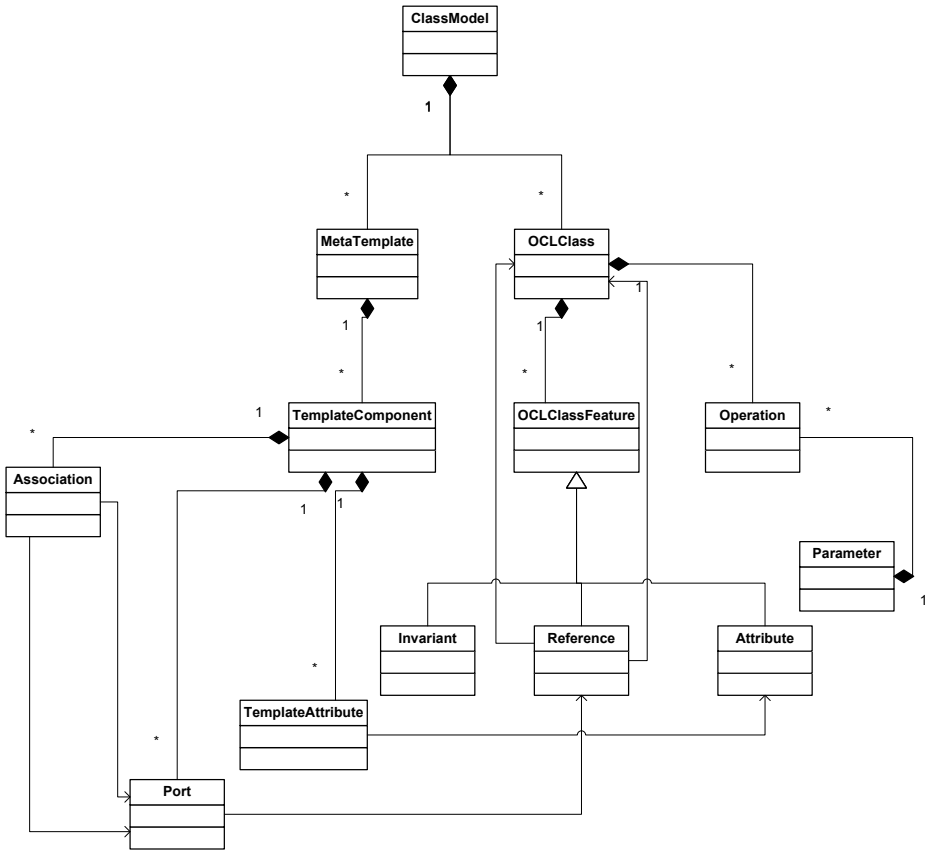
This section describes the data model used in the tool, which represents the internal structure used in the tool. The data model used in the Object modeler is designed on top of the data model in operation in the Class modeler. The data model is aligned with MOF [101], which is the metamodel of the UML [102]. This ensures the possibility of compatibility with other tools following the UML. Compatibility can be achieved by implementing model-to-model transformations [57], which can be realized fairly easy without the risk of information loss.

Following MOF simplified the usage of the Eclipse Modeling Framework (cf.

Section 7.3), as this framework is built to support MOF.

The data model additionally reflects several characteristics of the tool presented earlier in particular templates. The resulting structure will be discussed in the remainder of this section. First, the data structure of the Class modeler, visualized in Figure 7.25, will be addressed.

Figure 7.25: The data model of the Class modeler



The following description starts with the classes that can be found in the top layer of Figure 7.25 and then continues layer by layer.

ClassDiagram: This is the representation of the class diagram containing classes, templates and viewpoints.

ViewPoint: This is a coherent set of views consisting of subsets of classes, attributes and references that are included in the class diagram.

MetaTemplate: This class is an extension of TemplateComponent used to indicate that templates can be constructed on top of each other.

OCLClass: This is the tool's implementation of the UML Class concept. An OCLClass contains attributes and references to other OCLClasses.

TemplateComponent: Templates can consist of instances of classes and instances of other templates. The template component provides the top-level functionality for both types of components. It also provides the communication functionality for template components to interact with each other using ports and associations.

OCLClassFeature: Representation of the abstraction level for different structural components of the OCLClass structure, in accordance with the UML specification. The specializations include attribute, reference and invariant.

Operation: An operation can have multiple parameters and basically represents a combination of related OCL statements written to perform a specific task. It is used to increase modularity and usability in the model. Operation extends from the parameter class, as it is a specialization of the concept. Similar to parameters, operation also has a type, and it accepts multiple arguments/parameters as input.

Parameter: The parameter is a type of model object, and an OCL operation can have multiple parameters.

Association: The association represents the connection between two template components and contains connection ports for both ends.

Port: The port functions as an interface to a template component. If two template components need to interact with each other, they must have compatible ports. Each port represents a reference (connection between classes). To connect to the template components, they must have ports sharing the same reference object.

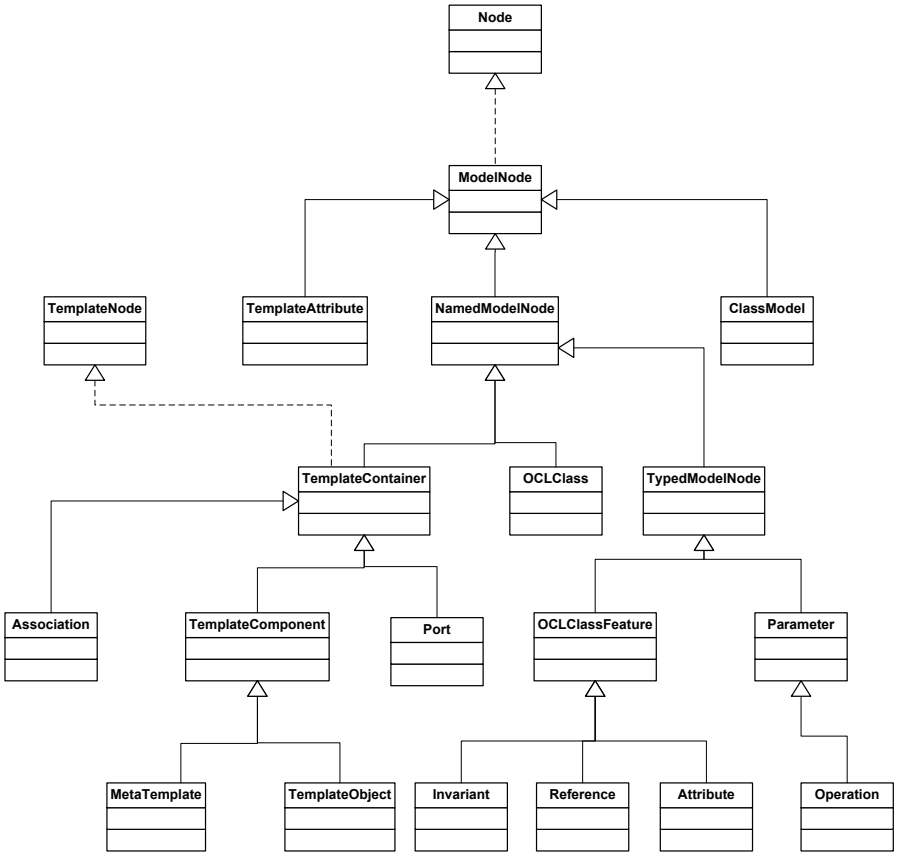
Invariant: Invariants help impose constraints on a model. This component is part of the data model to represent the invariant concept that is included in OCL [5].

Reference: References are the implementation of the association concept on a class level. This concept contains references to classes on both ends. References provide the functionality for defining multiplicities for both ends as well. References can be derived based on the result of OCL queries, supported by the Eclipse Modeling Framework's implementation of OCL.

Attribute: This class represents the implementation of the UML Attribute. TemplateAttribute: As described earlier (cf. Section 7.2), the tool allows the assignment of virtual attributes to templates that actually belong to elements included in the template.

Figure 7.26 illustrates the data model of the Object modeler.

Figure 7.26: The data model of the Object modeler



Node: This is the base interface for every element of the data structure. It provides generic methods for the publisher-subscriber design pattern [45] that is used in the application to synchronize the information between the visual representation of the model and the internal representation.

ModelNode: This is the implementation of the node interface and provides the basic concrete functionality of the property listeners following the publisher-subscriber design pattern.

TemplateNode: This class is an interface to provide extra functionality for templates, especially by extending them and querying their properties.

NamedModelNode: This class contains functionality to keep track of the model node being used in different templates. When an object is added to a template, the object registers itself to NamedModelNode. When the NamedModelNode is deleted in one of the created templates, then all instantiations will also be removed from the templates including it.

ClassDiagram: This class is the representation of the class diagram containing related classes and templates. It defines the concepts that the user of the Object modeler can make use of.

ViewPoint: This class represents the viewpoints that are specified as part of the analysis framework on the class diagram.

TemplateContainer: This class is responsible for the composition functionality of templates, allowing them to be implemented using already defined templates.

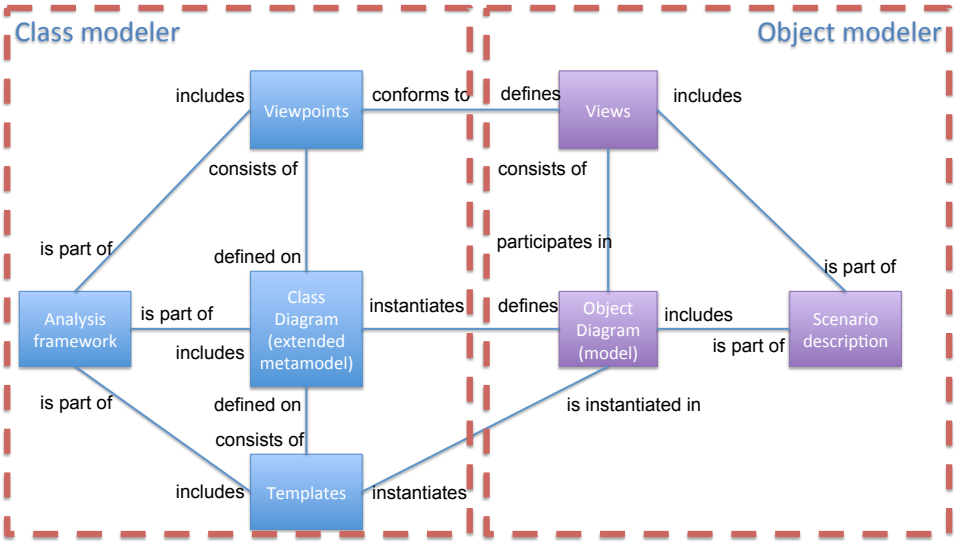
TemplateObject: This is an extension of the TemplateComponent class representing a leaf in a complex structure of templates specified based on other templates.

TypedModelNode: This represents the nodes that are part of a model and have a type. The types supported by the tool and OCL are integer, double and boolean.

The concepts TemplateAttribute, OCLClass, Association, MetaTemplate, Invariant, Reference, Attribute, Parameter and Operation have the same meaning for the Object modeler as previously described for the Class modeler.

The connection of the data model for Class and Object modeler is governed by the discussed tool (cf. Figure 7.27).

Figure 7.27: The connection between datamodel of Class and Object modeler



The Class modeler allows the creation of analysis frameworks. These frameworks include a class diagram, which is an extended metamodel. Viewpoints can be included in an analysis framework. Viewpoints are defined in the class diagram. Specifying templates based on the class diagram, as part of the analysis framework, is possible as well. The Object modeler component of the tool allows the creation of scenario descriptions based on a defined analysis framework. These scenario descriptions consist of an object diagram that instantiates the class diagram included in the analysis framework. In addition, views can be added to a scenario description, fostering the consideration of subsets of the object diagram. Views can be used based on the defining viewpoints.

Used 3rd party libraries

This subsection describes libraries provided by third parties that are used in the tool. These libraries were used in various parts of the tool with the aim of reusing existing, well proven and well documented functionalities. Especially in areas such as the visualization of large statistical datasets, the implementation of visual modeling editors and query language established solutions were reused. This helped to reduce the time between releases, ensured the quality of the tool and freed resources that could be used to focus on the tool's main purpose. In particular, this section describes the usage of the Apache Commons Mathematics Library, Eclipse modeling framework, JFreeChart library and JasperReports Library.

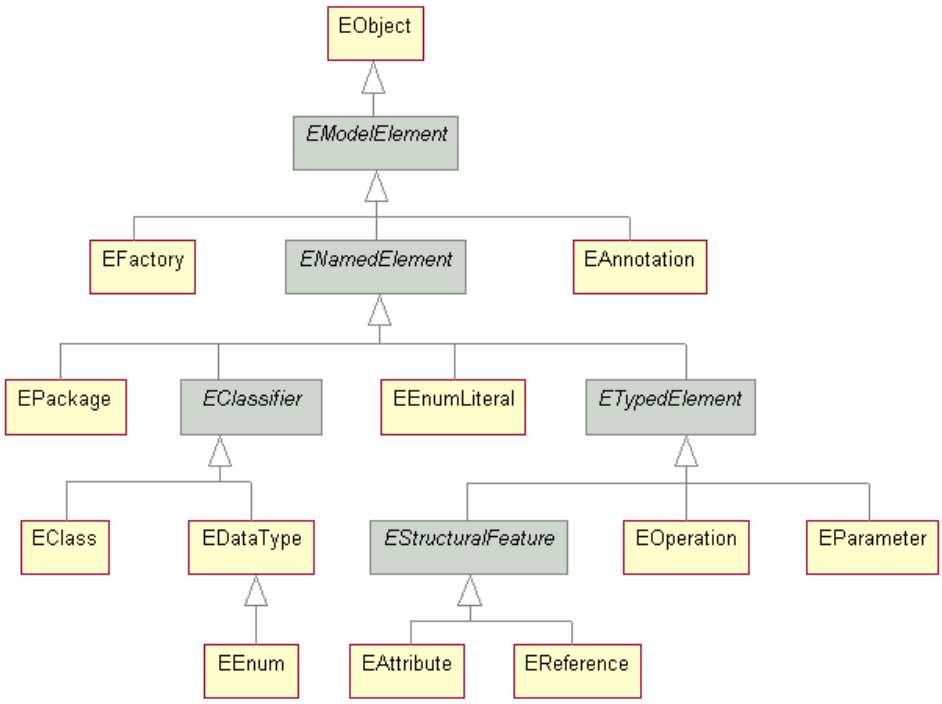
Apache Commons Mathematics Library

To implement the P2AMF plug-in, the Apache Commons Mathematics Library [73] was identified as a suitable library to realize the probabilistic distributions that P2AMF heavily makes use of (cf. Section 5.4). This open-source library supports more than 10 probabilistic distributions, including normal, Bernoulli and Weibull distributions. This library is well documented and therefore easy to integrate into other applications.

Eclipse Modeling Framework

The Eclipse Modeling Framework (EMF) [39] unifies Java programing, XML messages and UML modeling [250, 25]. It can be used to transform input provided in any of these three notations into the other two notations. The model followed to represent models in EMF is called Ecore [39]. EMF supports Essential MOF (EMOF) [249] as part of the OMG MOF 2.0 specification and follows a similar modeling hierarchy [25]. The type information of sets of instance models is defined in a so-called core model corresponding to the metamodel concept that can be found in EMOF. The metamodel for core models is the Ecore model containing the model elements, which are available for EMF core models in principle [24]. The core of Ecore is shown in Figure 7.28.

Figure 7.28: The structure of Ecore [75]



The Eclipse Modeling Framework features numerous editors, both graphical and textual, that can be used to create Ecore models and to perform transformations between them. In particular, the presented tool uses the offered plugins for the visual creation of models in both the Class and the Object modeler.

As part of the EMF Model Development Tools, EMF offers Eclipse OCL, which implements the OCL language. Eclipse OCL can be used to specify the pre- and post-conditions on Ecore and UML models and to evaluate queries on them. In particular, it can be used to evaluate attribute values while considering the state of the model.

This characteristic was used to implement the P2AMF plug-in of the tool. Eclipse OCL also features a console for the interactive evaluation of OCL expressions. This console was used in the present tool to offer the user of the Object modeler the capability to investigate the model based on user-defined queries. Moreover, Eclipse OCL features mechanisms to validate OCL code. As described in the feature section of the Class modeler (cf. Section 7.2), this is a helpful feature to ensure code quality. Therefore, the tool makes use of this functionality.

jFreeChart

To visualize the analysis results and, in particular, to draw histograms based on the outcome of sampling, the tool uses the free Java chart library JFreeChart [92]. This library supports the drawing of a variety of charts and allows the configuration of the visualization based on the use case.

JasperReports Library

In the current implementation, an export functionality to generate PDF reports describing the analysis of the architecture models was included. This function was realized using the open-source reporting engine JasperReports Library [113]. Using JasperReports Library, one can develop applications that generate reports and dynamic content. The library allows these reports to be exported in a number of files, including PDF, Microsoft Word, XML and RTF.

7.4 The areas of contribution in relation to the presented artifact

In part of the introductory chapter (cf. Section 1.2), the contribution of the author to the development of the tool's features was described. This section concludes the description of the artifact by relating the areas in which work was performed to the sections of this chapter covering them. The features of the tool will be discussed following the structure of Figure 1.4 and Figure 1.5, starting with a discussion of the work performed in the analysis area and then considering the modeling aspect.

The tool's functionality to update the analysis framework was described in Section 7.2.

The capabilities of the tool to perform impact analysis are covered in Section 7.3. The input configuration of the tool was covered in the section describing the user interface (cf. Section 7.1). The tool's inference engine was described in the Sections 7.2 and 7.3. In addition, research related to this feature was already covered in the section describing the underlying design decisions that led to the implemented inference engine (cf. Section 5.4).

The actual specification of an analysis framework is a manual task that is not directly reflected by the presented tool (cf. Sections 7.1). The tool provides support for translating a knowledge base, i.e., the theoretical foundation of the analysis of a system property, into an analysis framework. This can be achieved using the tool's user interface (cf. Section 7.1) connecting the user input to the internally used data structure (cf. Section 7.3).

The tool's functionality to specify templates is covered in Section 7.2. The tool's support for the usage of specified templates is discussed in this section too (cf. Section 7.2). The tool's functionality to automatically instantiate class diagrams included in analysis frameworks is reflected in Section 7.2. The user interface was described in Section 7.1. The author's work on the usability topic is reflected in the

section describing the user interface (cf. Section 7.1) and in the section describing the usage of views and viewpoints (cf. Section 7.2). The outcome of the author's and other contributors' work to address the visualization of the analysis results can be found in Section 7.2. The tool's support for the creation and editing of models was described in the section covering the Object modeler's workflow (cf. Section 7.1) and the user interface of the Object modeler was described in this section as well (cf. Section 7.1). Manual model creation is indirectly supported by the user interface (cf. Section 7.1).

Area	**The author's contribution**	**Discussed in section**
Analysis	Update of Analysis Framework	• Distinct functionality of the Object modeler/Update of the used analysis framework
	Impact Analysis	• Distinct functionality of the Object modeler/Sources of impact tracing
	Input Configuration	• User interface
	Inference Engine	• Distinct functionality of the Class modeler/P2AMF validation • Architecture of the tool/The resulting tool architecture • Design Decision IV: Inference engine
	Probabilistic Reasoning	• Design Decision IV: Inference engine

Continued on next page

Table 7.2 – continued from previous page

Area	The author's contribution	Discussed in section
	Knowledge Base	• n/a
	Actual Specification of Analysis Framework	• User interface • Architecture of the tool/The used data model
Modeling	Templates	• Distinct functionality of the Class modeler/Complexity reduction using templates • Distinct functionality of the Object modeler/Complexity reduction using templates
	Automatic Model Instantiation	• Distinct functionality of the Object modeler/Automatic instantiation of the class diagram
	User Interface	• User interface
	Usability	• User interface • Distinct functionality of the Object modeler/Creation of filtered views based on viewpoints

Continued on next page

Table 7.2 – continued from previous page

Area	The author's contribution	Discussed in section
	Result Visualization	
		• Distinct functionality of the Class modeler/Result visualization
	Model Creation & Editing	
		• User interface
	Actual Model Creation	
		• User interface

Table 7.2: Mapping between the areas in which work was performed and the sections describing the outcome of this work

Chapter 8

Demonstration of the usability of the tool

Following the applied research method, this chapter continues with the demonstration of the developed artifact (cf. Chapter 2). Unlike the previous chapter, which dealt with the demonstration of the technical aspects, this chapter describes the usability of the tool. The usability of the tool is described by discussing different use cases in which the tool has been applied. A theoretical demonstration of the tool without discussing practical applications of the presented tool was not considered to be equally meaningful.

First, the usability of the tools component to specify the Class modeler for the analysis of Enterprise Architecture models is presented. Next, a subsection reports on the tool usage by practitioners and mainly non-theory experts to analyze real or realistic cases. This demonstrates the usability of the Object modeler.

8.1 Usage of the tool to specify theory

This section reports on the usage of the tool to specify analysis frameworks and the verification of the specified theory. More than 10 different analysis frameworks have been specified over a 6-year period. Using the tool for the specification of analysis frameworks was especially helpful in identifying the requirements of the Class modeler. It also contributed to the validation of the implementation of the inference engine, as many authors of frameworks have started to implement simple test cases. The calculation results for those test cases could then be compared to manual calculations to verify that inference was performed properly.

During the presented research, it was beneficial that the developers of the different frameworks were employed at the same department as the author, fostering spontaneous assessments of the tool.

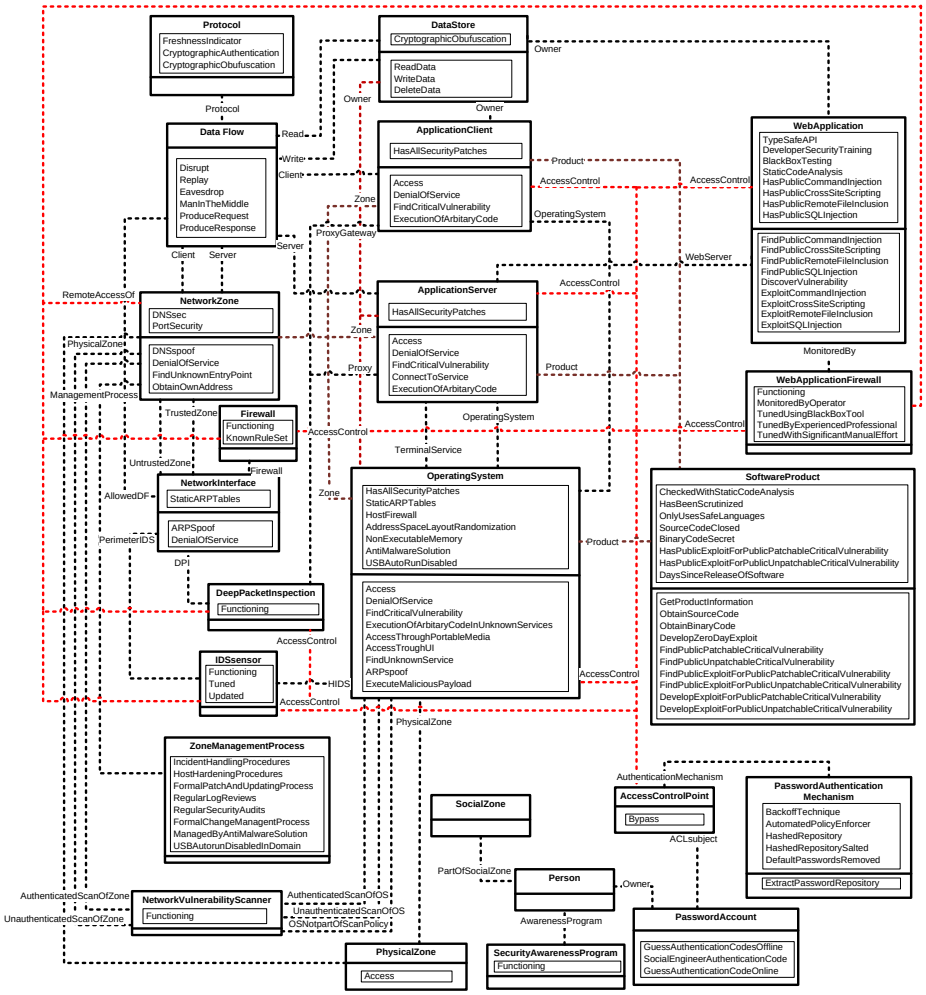
8.2 Specification of a cyber security analysis language

In [248], a language for conducting cyber security analysis (CySeMol) is presented. This framework is an extended version of the framework presented in [246]. Both frameworks were implemented using the PRM formalism, the only inference engine supported at that time. The framework presented in [246] consists of more than 20 concepts that are relevant for cyber security analyses. These concepts are as follows:

- Access Control Point
- Application Client
- Application Server
- Data Flow
- Data Store
- Deep Packet Inspection
- Firewall
- IDS Sensor
- Network Interface
- Network Vulnerability Scanner
- Network Zone
- Operating System
- Password Account
- Password Authentication Mechanism
- Person
- Protocol
- Security Awareness Program
- Software Product
- Social Zone
- Web Application
- Web Application Firewall
- Zone Management Process

The authors of [246] describe these concepts with regards to whether they are assets, possible attacks or defense mechanisms. By applying CySeMoL, a user creates an attack graph (cf. Section 5.6) that can be used to analyze the likelihood of a successful attack. The analysis framework is visualized in Figure 8.1.

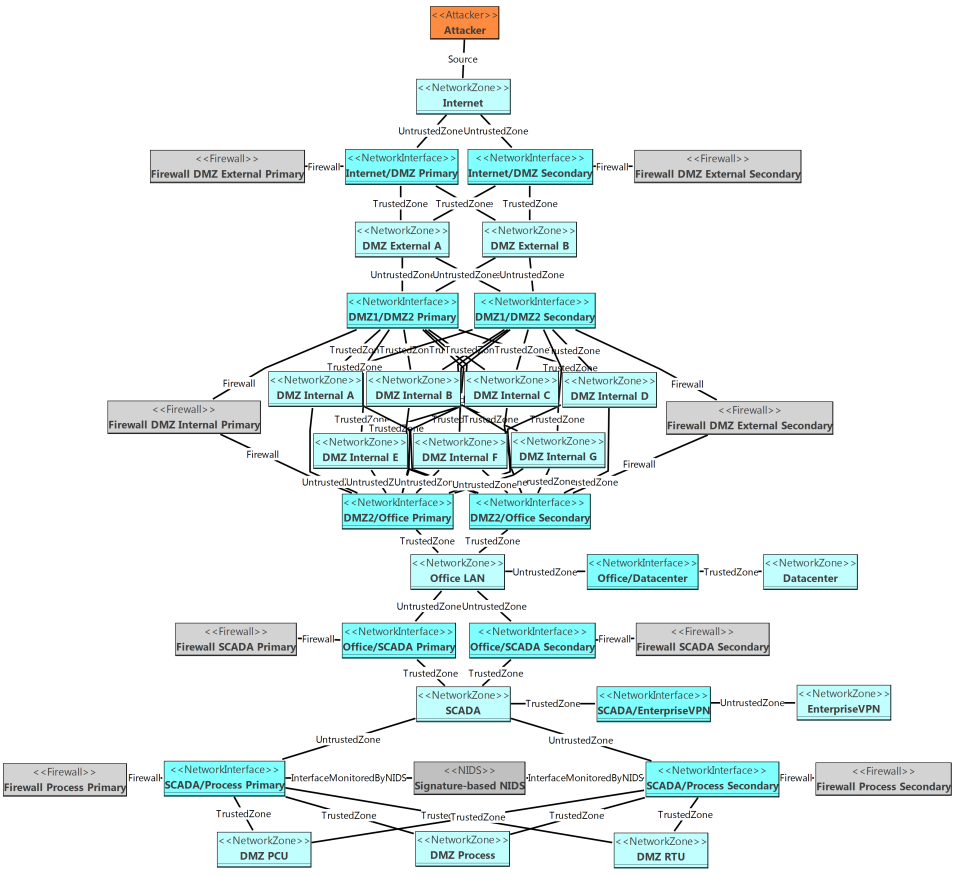
Figure 8.1: The Cyber Security Analysis Language (CySeMol)



The authors of CySeMoL validated it using the tool in four case studies conducted in the power domain and at a large infrastructure provider. These case studies led to four enterprise models, two consisting of 50 to 100 objects, one with 100 to 200 objects and one with more than 500 modeled objects. The reworked version of CySeMoL, presented in [119], was verified in a case study conducted in collaboration with an electric power company. In this study, a model with more than 100 objects was created. The entire object diagram consisted of 143 assets, 571 attack steps, 341 defenses and 1780 connections between attack steps. The

likelihood of an attacker on the internet obtaining administrator access to one of the modeled business systems was among the possibilities analyzed. It was identified that the likelihood increased significantly in relation to the effort applied by the attacker. If an attacker spent one hour on this endeavor, his or her likelihood of success, as defined by obtaining administrator access, would be 8 percent. An investment of 40 hours would increase the likelihood to 85 percent. Finally, spending 80 hours would lead to a likelihood of 98 percent. A subset of this model is visualized in Figure 8.2.

Figure 8.2: An anonymized subset depicting the network topology of the studied enterprise (where DMZ means demilitarized zone).



8.3 Other analysis frameworks

In [186], a framework for the analysis of business performance and organizational structure was presented. The author of the framework used the tool to implement it. Additionally, she validated the framework using three case studies in collaboration with the automotive industry. This resulted in three models that were implemented using the tool, each consisting of 200 to 500 modeled objects.

The authors of [182] used the presented tool to specify a framework for the enterprise-wide analysis of service availability. This framework was implemented based on the PRM formalism. Once the framework was in place, the authors conducted four case studies: one within the financial sector, one at a travel agency and one at a company that preferred to remain anonymous. This resulted in five applications of the tool, leading to five models with approximately 50 objects each.

In [180], a combined framework for the analysis of several system properties is presented. This framework makes use of the insights gained during the implementation of frameworks in PRM but is based on P2AMF. The framework presented in [182] was later further developed into an even more powerful framework [136]. A verification of the availability component of the framework in collaboration with the financial industry was presented in [209]. The conducted availability analysis resulted in a model consisting of 201-500 modeled objects.

[260] presents a framework for the analysis of enterprise-wide interoperability that was specified using the tool. This framework was implemented based on P2AMF. It was verified, using the tool, in five case studies that were carried out in the power domain, health care sector and defense sector. In total, five models, two with fewer than 50 modeled objects, two with 50 to 100 model elements and one with 200 to 500 objects, resulted from this verification endeavor.

In [205] , a framework for the analysis of application modifiability was presented. This framework was implemented in the tool. The framework was validated using the tool once again, this time in an educational institution. The resulting model contained between 200 and 500 objects [223].

A framework for the analysis of data accuracy was implemented in the tool based on that presented in [184]. This framework was then further developed and, similar to [182], integrated into a combined framework [182] . After another iteration of fine-tuning, it was even added to [136].

[183] presents a framework for the prediction of application usage. This framework and its successor, presented in [182] were implemented in the tool. To valid the frameworks, five case studies were conducted, and the results were implemented in the tool.

A framework for the analysis of enterprise profitability was presented in [263] and [134]. Yet again, this framework was implemented in the tool. The tool was also used to validate the framework in a case study and resulted in a model with 50 to 100 modeled objects.

To compare the analysis results for numerous system properties, [204] illustrated the usage of utility theory. This was implemented using the tool for Enterprise

Architecture analysis as well, both as a stand-alone framework and as part of [136].

As already mentioned, a combined framework for the integrated analysis of application modifiability, data accuracy, application usage, service availability, interoperability, cost and utility was presented in [136]. This framework was also implemented in the tool.

The described tool applications are summarized in 8.1.

Considered system property	**Publication**	**Number of included entities**	**Model(s) based on case studies used for validation**
Cyber security	[246, 119]	23/22	• 2* 50 to 100 entities • 2* 100 to 200 entities one model • 1 * more than 500 entities
Business performance and organizational structure	[186]	13	• 3* 200 to 500 entities
Service availability	[182]	14	• 5* 50 entities
Enterprise interoperability	[260]	12	• 2 * less than 50 entities • 2 * 50 to 100 entities • 1 * 200 to 500 entities

Continued on next page

Table 8.1 – continued from previous page

Considered system property	Publication	Number of included entities	Model(s) based on case studies used for validation
Application modifiability	[205]	15	• 1 * 200 to 500 entities
Data accuracy	[184]	6	• Validated using test cases instead of case studies
Application usage	[183]	8	• 5 case studies
Enterprise profitability	[263, 134]	14	• 1 * 100 to 200 entities
Utility theory applied on service availability and cost	[204]	12	• Validated using test cases instead of case studies

Continued on next page

Table 8.1 – continued from previous page

Considered system property	Publication	Number of included entities	Model(s) based on case studies used for validation
Service availability, Data accuracy, Service Response Time, Application usage	[180]	24	• Validated using test cases instead of case studies
Application modifiability, data accuracy, application usage, service availability, interoperability, cost, and utility	[136]	29	• Validated using test cases instead of case studies

Table 8.1: Implemented analysis frameworks

8.4 Usage of the tool to perform analysis

The tool was also presented to the second intended user group. This group consists of typical Enterprise Architecture tool users who want to use the tool to perform architecture modeling and analysis and lack interest in developing analysis frameworks themselves. These typical Enterprise Architecture tool users were students taking courses in the author's department or industry practitioners.

Over a period of three years and spread over three courses, a total number of 29 students used the tool to perform analyses of different scenarios. The students made use of the frameworks mentioned in the previous section. Mostly [136] was used; however, in one course, [260] was applied by the students. The students taking that course were typically studying various fields of engineering and had in common that they were in their penultimate or final year of education.

The students had a good understanding of the analysis frameworks and the overall approach. Collaborating with them helped identify potential improvements in the usability in terms of workflow, user interface, tool documentation and model visualization.

The students conducted 10 case studies in groups of two to four students. They worked closely with Swedish and international companies, including insurance providers, energy and automation companies, providers of air and railway technology and aerospace and defense companies. The case studies included a scoping of the project, data collection for Enterprise Architecture analysis, analysis, result interpretation, suggestions of alternatives and to-be scenarios as well as result presentation.

The tool was also presented to practitioners involved in IT decision-making as part of their job. This audience was taking an evening course on the topic of Enterprise Architecture as a means for decision-making in the author's department. As part of the course, the participants used the tool to conduct analyses in their field of their daily work. The practitioners used the tool both on their own and under the supervision of the author and his colleagues. A total of 20 participants used the tool to conduct analyses, most of whom had a background in the financial industry, defense sector, telecommunication sector or governmental sector. The practitioners used the analysis framework presented in [136]. They often focused on the analysis of availability, data accuracy or application modifiability.

Presenting the tool to this audience helped assess the tools from an end-user perspective. Some of the practitioners were experienced users of other Enterprise Architecture tools and could therefore provide valuable suggestions with regards to the functionality that they considered to be lacking. This group was also very conscious regarding the time spent creating a model in the tool. Therefore, several suggestions were made that helped to accelerate the tool's workflow and support the automatic generation of models.

The described applications are summarized in 8.2.

User group	Number of users	Year
Students	7	2011
Students	8	2012
Students	14	2013
Practitioners	4	2012
Practitioners	16	2013

Table 8.2: Users of the presented tool performing analysis

Chapter 9

Evaluation

In this chapter, an evaluation of the tool is presented. Specifically, whether the Enterprise Architecture analysis tool actually fulfills the objectives it was meant to fulfill is investigated. Following the used research method (cf. Chapter 2), the requirements described in Chapter 4 are used to perform this evaluation. The final section of Chapter 4 (cf. Section 4.4) summarizes the requirements derived from Enterprise Architecture tool evaluations, the supported Enterprise Architecture analysis method and theoretical thoughts on CAD tools and expert systems. The fulfillment of these requirements is dependent on the design decisions made, which are described in Chapter 5. This chapter is separated into two parts. First, the presented tool will be evaluated against each of the nine elicited requirements (cf. Section 4.4). Second, a general evaluation will be presented. The fact that the tool focuses on the analysis of Enterprise Architecture models is reflected upon as well. The evaluation of the criterion analysis is performed comprehensively and in detail. Before considering the nine elicited requirements, it is important to understand the context of the evaluation: neither the specified analysis frameworks nor the models created by the tool users to perform analyses are the focus. Instead, the evaluation focuses on the tool as such, decoupled from a specific analysis framework or a selected scenario that should be analyzed with regards to a certain system property. It was the intent that the tool should be usable; capable of performing proper analysis; have administration and presentation capabilities; feature an extendable metamodel; support importing, editing, and validating of external data; feature a repository; offer an expressive metamodel; and support modeling independent of its input. The goal of this chapter is therefore to evaluate whether this endeavor was successful.

9.1 The tool shall offer a high degree of usability

[214] suggests evaluating the usability of software artifacts according to five dimensions: effectiveness, efficiency, level of engagement, error tolerance and ease

of learning. To evaluate the usability of the tool presented in this thesis, a questionnaire was used and sent to the different user groups of the tool. In total, 35 samples were collected. Ten of the respondents were theory experts using the tool to specify analysis frameworks, 15 were practitioners, and 12 were students who used the tool as part of their education to both define analysis frameworks and use these frameworks during case studies.

The answers received from the theory experts describe to a large extent how this group experienced the usability of the Class modeler. This was the tool component that this group typically used the most, mainly to specify analysis frameworks. However, as the members of this group typically performed case studies to validate the created analysis framework in practice, this group also made use of the Object modeler. Therefore, the answers received from the members of this group can be considered to provide a complete picture of the usability of the tool resulting from the research work described in this thesis.

The practitioners only used the Object modeler component. The members of this group only applied the tool to perform analyses. They never specified or even studied the used analysis frameworks. Their answers regarding their perception of the usability are therefore only valid for the Object modeler.

The students used both components of the discussed tool. As parts of their coursework, they specified and modified analysis frameworks and applied these frameworks to evaluate Enterprise Architecture models. Their perception of the usability therefore covers both components of the tool.

In total, seven claims were used to evaluate the five previously mentioned dimensions. These claims are visualized in Table 9.1. A five-point Likert scale ranging from 1 ("strongly agree") to 5 ("strongly disagree") was used. An additional category ("no opinion") was added to reduce arbitrary answers. It must be mentioned that the seven claims were each followed by a short set of instructions indicating that the answers should be given based on the individual background of the user. Thus, theory experts were to express whether they agreed with the claims to evaluate the usability with regards to the specification of analysis frameworks. On the other hand, practitioners were to give their opinion on the same claims with the goal of evaluating the usability of the tool from a scenario-analysis perspective. This approach allowed the reuse of the same claims for all user groups, comparison of the answers and the synthesis of a holistic perspective of the usability.

A final question was included to ensure that the respondents were familiar with the latest version of the tool and were not describing the usability of one of the prototypical predecessors. This question was "When did you use KTH's Enterprise Architecture analysis tool most recently", with the possible answers of 2011, 2012, 2013 and 2014.

The questionnaire was realized as an online survey. During a 10-day period, answers could be submitted, and a reminder was sent after the first seven days. In total, 12 theory experts, 28 practitioners and 22 students were asked for their opinion.

The questionnaire can be found in the appendix (cf. Appendix).

Dimension	Definition	Question number	Claim
Effectiveness	The completeness and accuracy with which users achieve specified goals	1	My goals for the Enterprise Architecture Analysis Tool were met
		2	I received the results I expected from the Enterprise Architecture Analysis Tool
Efficiency	The speed with which work can be done	3	I completed the task within the Enterprise Architecture Analysis Tool quicker compared to using other tools that I have at my disposal
Level of Engagement	How pleasant, satisfying or interesting an interface is to use	4	I had a pleasant experience with the Enterprise Architecture Analysis Tool when using it
Error Tolerance	How well the product prevents errors and helps the user recover from any errors that do occur	5	The user interface of the Enterprise Architecture Analysis Tool helped me avoid making errors
		6	I was able to recover when I made an error
Ease of Learning	How well the Enterprise Architecture analysis tool supports both initial orientation and deeper learning	7	The user interface of the Enterprise Architecture Analysis Tool was predictable

Table 9.1: The claims used to evaluate the usability

The answers to claim 1 are depicted in Figure 9.1. It can be seen that more than two-thirds (26 out of 35) of the respondents agreed or strongly agreed that the tool met their goals. In addition, eight answers of "neutral" were received, two of disagreement and one of no opinion.

Considering Figure 9.2, it can be identified that more than two-thirds (26 of 37) of the users agreed or strongly agreed that they received the results that they expected from the tool. Furthermore, eight participants responded with "neutral", two respondents disagreed with the statement, and one selected "no opinion".

Figure 9.1: The answers for claim 1

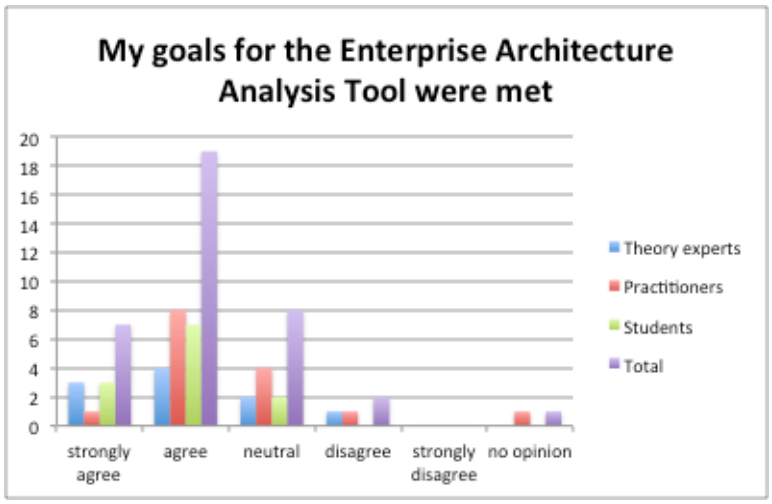
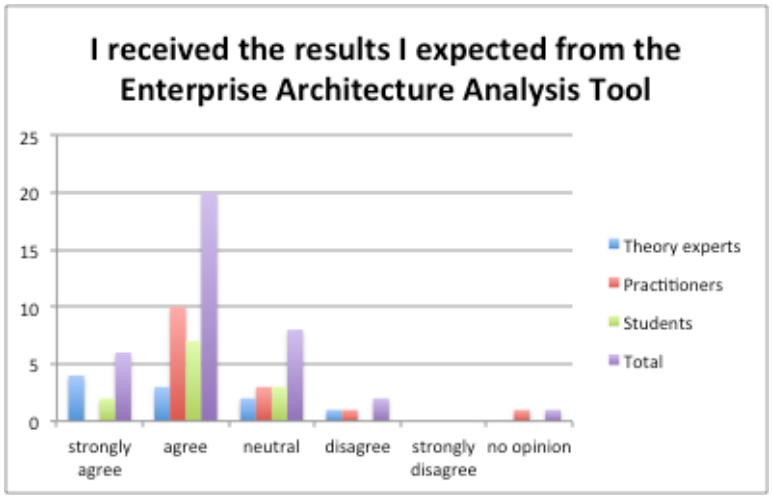


Figure 9.2: The answers for claim 2



In Figure 9.3, the answers for the third claim are visualized. Here, thirteen respondents agreed or strongly agreed that they completed their tasks faster using the tool discussed in this thesis compared to other tools. Five respondents answered "neutral", seven "disagree" and twelve "no opinion".

Fourteen respondents agreed and three strongly agreed that they had a pleasant experience using the tool. Thirteen respondents chose to answer "neutral" for this

Figure 9.3: The answers for claim 3

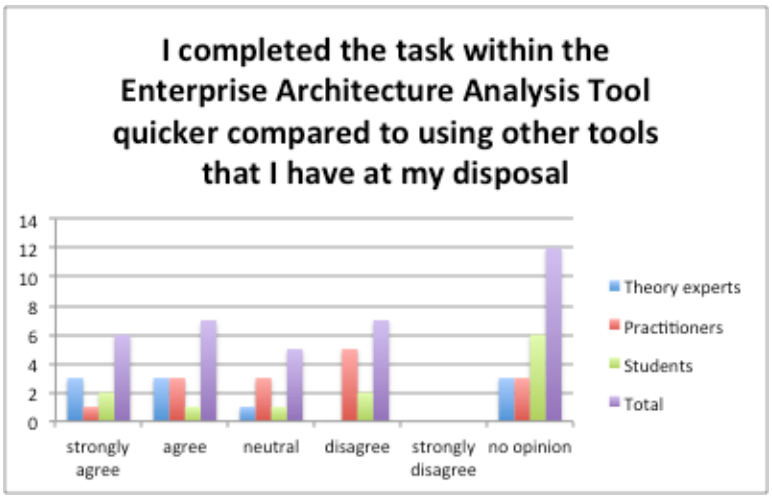
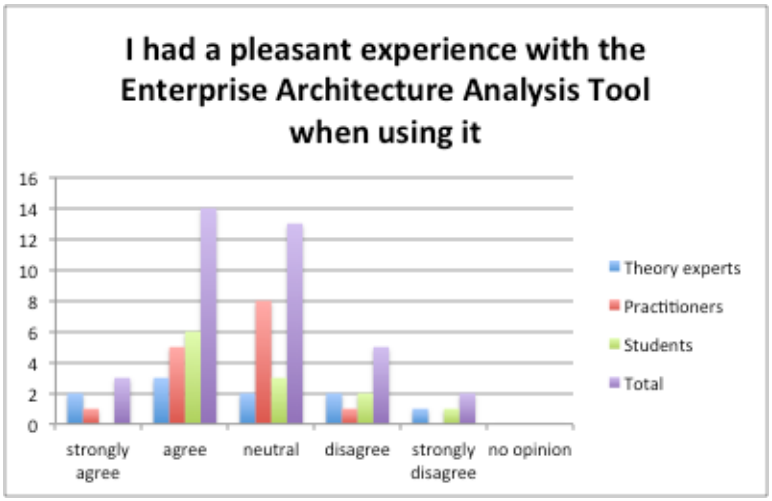


Figure 9.4: The answers for claim 4



claim (cf. Figure 9.4). Seven respondents disagreed or strongly disagreed that they had a pleasant experience.

The answers to claim 5 are depicted in Figure 9.5. Here, it can be seen that thirteen respondents either agreed or strongly agreed that the user interface helped them avoiding errors. Thirteen respondents answered “neutral”, nine disagreed or strongly disagreed, and two had no opinion.

Figure 9.5: The answers for claim 5

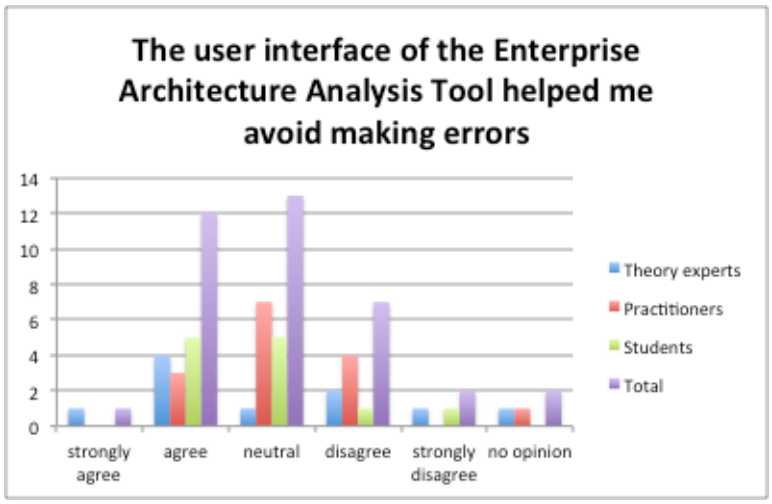
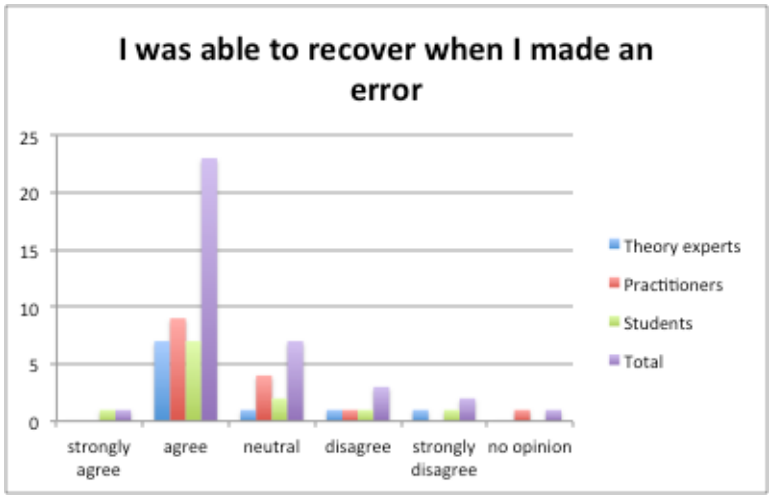


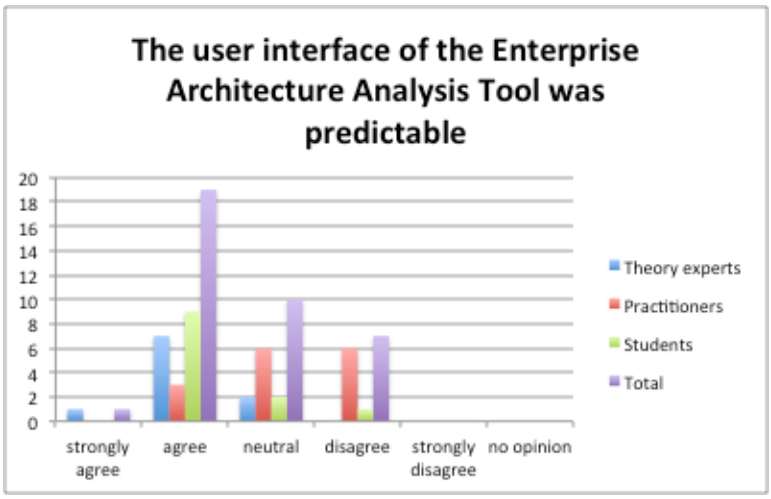
Figure 9.6: The answers for claim 6



In total, twenty-four respondents either agreed or strongly agreed that they were able to recover after making an error (cf. Figure 9.6). Seven of the respondents answered "neutral", five answered "disagree" or "strongly disagree", and one answered "no opinion".

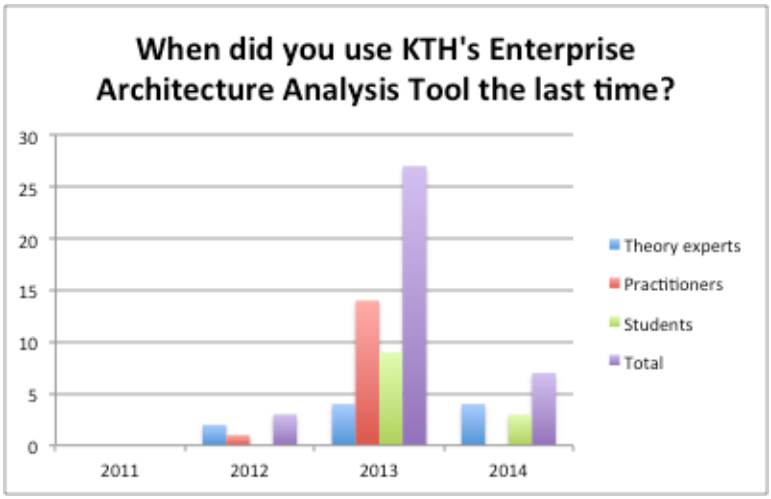
In total, nineteen of the respondents either agreed or strongly agreed that the tool interface was predictable (cf. Figure 9.7). The answer "neutral" was given by

Figure 9.7: The answers for claim 7



ten respondents, and disagreement was expressed by seven respondents.

Figure 9.8: The answers to the control question



The collected answers to the previously mentioned control questions are visualized in Figure 9.8. Three questionnaires were submitted by users who most recently applied the tool in 2012. The tool was most recently used in 2013 by twenty-seven respondents and in 2014 by seven participants.

Based on the answers given to the control question, all collected samples were relevant and based on the latest version of the tool. The answers given in response to the first claim (cf. Table 9.1) indicated that most users consider the tool to be effective, or at least not ineffective, with thirty-four respondents either in agreement with or neutral towards the claim "My goals for the Enterprise Architecture analysis tool were met". The experienced effectiveness or the absence of experienced ineffectiveness is also reflected by the answers to the second claim. In total, thirty-four participants in the survey agreed with or were neutral toward the claim "I received the results I expected from the Enterprise Architecture analysis tool".

Thirteen respondents indicated that the tool is efficient in response to the third claim. Twelve did not an express an opinion of the claim "I completed the task within the Enterprise Architecture analysis tool more quickly than I would have using other tools I have at my disposal".

With regards to the fourth question, thirty participants reported having had a pleasant or a not unpleasant experience. These answers attest that the tool provides a satisfying level of engagement.

Most of the asked tool users expressed that they were satisfied or at least not dissatisfied by the tools error tolerance. They also indicated that they appreciated how the tool prevents errors and helps them recover from errors made during the application of the tool. In total, twenty-sex tool users either agreed or did not disagree that the tool helped them avoid making errors. Furthermore, thirty-one users reported being able to recover after making an error.

Only seven respondents reported that the user interface was unpredictable; in comparison, thirty expressed their agreement with or neutrality toward the claim "The user interface of the Enterprise Architecture analysis tool was predictable". Based on these answers to claim seven, the tool can be considered to be fairly easy to learn.

In conclusion, considering all seven claims, it can be realized that the users generally perceive the tool to be usable or at least not unusable.

As described in the beginning of this section, the answers received from the practitioners only describe their experience with the Object modeler; this group did not use the Class modeler component of the tool. However, the other two groups, theory experts and students, utilized both components and were therefore capable of judging the usability on a holistic level. Regarding the seven evaluated claims, the answers received from the practitioners never significantly deviated from those received from the other two groups. Thus, the perceived usabilities of the two components are fairly similar.

During development, a clear user interface with intuitive menu navigation was sought. The screenshots included in Chapter 7 show how this concern was addressed.

From a technical perspective, usability was also supported, as well-known and frequently used development platforms were used. The selected platform (Eclipse Rich Client Platform) is widely known and integrates into the user's desktop envi-

ronment. Therefore, he or she experiences the application as being, from a visual perspective, similar to the other software programs that he or she uses.

Additionally, a structured workflow was developed, also presented in Chapter 7, which is covered by the tool. Following this workflow, the usage of the tool is eased. Additional support material in terms of a manual, a tutorial and interactive screencasts illustrating the usage of the tool were created as well.

The fact that students, practitioners and theory experts were able to use the tool without help (cf. Chapter 8) also provides evidence that the tool has a certain level of usability. Table 9.2 summarizes the different user groups.

Users	Task
14	Theory specification
49	Analysis of scenarios
28	Education using fictitious scenarios

Table 9.2: User groups applying the tool

9.2 The tool shall possess analysis capabilities

The presented tool was developed with the goal of supporting the analysis of Enterprise Architecture models. Therefore, it was equipped with several features to support analysis. Even more importantly, analysis of architecture descriptions was not considered as an add-on but is deeply reflected in the tool's architecture. The tool has one separate component, the Class modeler, with the single task of supporting the specification of analysis frameworks. Analysis frameworks can be specified based on P2AMF (cf. Section 5.4), allowing the expression of the impact of attributes on one another in terms of calculation rules. The Class modeler allows specifying how the analysis results should be visualized. This feature also contributes to the usability of the tool. The Object modeler, the second tool component, was developed with the goal of supporting the usage of the previously specified analysis framework. The Object modeler features a probabilistic inference engine, allowing the values of attributes to be derived based on provided evidence [235](cf. Section 5.4). This inference engine can also be used to conduct impact analysis with the goal of identifying the cause of a certain attribute value (cf. Section 7.2).

Another feature connecting the Class modeler and Object modeler and supporting analysis at the same time is the Object modeler's ability to update analysis frameworks. As described in Section 7.2, the Object modeler allows a given object diagram to be updated to the latest version of an analysis framework without the need for recreating the model.

As reported in the demonstration chapter (cf. Chapter 8), the tool has been used to specify roughly a dozen analysis frameworks, which have been used to

analyze more than 50 scenarios. The considered system properties are summarized in Table 9.3 , whereas Table 9.4 reports on the conducted analysis.

Considered system property
Cyber security
Business performance and organizational structure
Service availability
Enterprise interoperability
Application modifiability
Data accuracy
Application usage
Enterprise profitability
Service Response Time
Cost

Table 9.3: Evaluated properties

Number of conducted analysis (on real scenarios)	Size of the models
25	<50 entities
6	50 to 100 entities
4	101 to 200 entities
14	201 to 500 entities
1	>500 entities

Table 9.4: Number of conducted analyses

Supporting analysis with functionality and in terms of the tool architecture is not enough. It must also be ensured that the analysis is conducted properly. Keeping in mind that analysis should be performed independently of the used class diagram or created object diagram, general test cases were created. These test cases were meaningless from an Enterprise Architecture analysis perspective. However, they were extremely valuable from a development perspective. The test cases described neither analysis frameworks nor scenarios of interest. Instead, fictitious, controllable subsets of typical class diagrams and instantiating object diagrams were used to evaluate whether the analysis-related tool functionality and particularly the inference engine (cf. Section 5.4) of the tool worked as expected.

The test cases covered the following aspects in particular, presented in the order of usage as described in the previous chapter (cf. Section 7.1):

1. Specification of attribute derivation
2. Specification of invariants
3. Modeling of objects
4. Relating of objects
5. Provision of data for object attributes
6. Configuration of sampling
7. Evaluation of the model
8. Modification of the model

The steps that were not considered with test cases were preparations of the analysis that could be evaluated by visually considering the tool, including the modeling of classes, relation of classes and relevant attributes. Here, it is possible to determine whether the tool works properly by comparing the visual result to the model that the user intended to create. The tool works properly if a class is created, a relation between two classes is created and attributes are added to a class when the user intends to perform each respective action.

The other steps of the workflows that were not covered using the test cases are the steps that do not contribute to Enterprise Architecture analysis from a tool perspective but are manually carried out by the user, namely, the understanding of the real situation and interpretation and comparison. These steps cannot be directly performed by the tool; instead, the tool user has to ensure that he or she understands the considered situation properly. He or she can receive support from the creator of the used class diagram, as it can incorporate uncertainty (cf. Section 3.2). The interpretation and comparison step addresses the comparison of different analyzed scenarios and the decision-making based on this comparison. This step is not directly supported by the tool; instead, it needs to be performed manually.

To determine whether the step specification of attribute derivation was carried out properly, numerous test cases consisting of various numbers of classes with one or more attributes were created. Next, attribute derivations were assigned to the included attributes. In this step, whether the attribute specification was stored correctly so that it could be used in the later steps of the workflow was evaluated, both in terms of instantiation (the modeling of objects step) and inference of attribute values (the evaluation of the model step). Proper storage was evaluated directly using a debugger and indirectly by comparing the outcome of the later steps to the included attribute specifications. Finally, it was also checked whether the Class modeler would allow the use of incorrect OCL code for the specification of attribute derivations.

During this evaluation, it was found out that this step was carried out correctly for the used test cases. Proper OCL code was stored to be used in the Object modeler. The tool also identified the erroneous specified attribute derivation contained in the test cases and informed the user.

To evaluate the step specification of invariants, it was investigated whether invariants were stored correctly so that they could be used in the later steps of

the workflow (the evaluation of the model step). Yet again, the proper storage was evaluated directly using a debugger and indirectly by comparing the outcome of the later steps to the included attribute specifications. As for the specification of attribute derivations, it was also investigated whether the tool only accepted well-formed OCL code.

Based on the results, this step was performed properly for the used test cases. The invariants included in the test cases were saved as they were specified if the specifications followed the OCL syntax.

The proper performance of a step-by-step modeling of objects was considered by evaluating the handling of model creation in the Object modeler. Specifically, the instantiation process of changing class diagrams into object diagrams was evaluated.

An investigation was performed on whether the tool ensures that the object diagrams are valid instantiations of the used class diagrams, i.e., if only the instantiations that are in accordance with the used class diagrams are possible.

For the test cases that were used, it was realized that the tool allowed the creation of only those objects that were previously defined for the used class diagram. It was not possible to create objects that did not have a foundation in the class diagram. The instantiated objects reflect the class diagrams correctly because they possess the described attributes, including the specified attribute derivations.

To evaluate the step that is related to the objects, the author investigated whether it was possible to relate the objects based on the used class diagram. In addition, the tool's capabilities were challenged by relating objects even when the class diagram did not feature a connection between them. Additionally, the capability for handling multiplicities was evaluated. An attempt was made to create models that either exceeded the multiplicities that were specified in the class diagram or undercut them. The result of this evaluation was that for the used test cases, the Object modeler allowed only the creation of object diagrams that followed the used class diagram. The Object modeler identified all of the discrepancies that were tested, and it reported them to the user in such a way that he or she could correct the object diagrams.

After having evaluated the two-step modeling of the objects and the relating of the objects, the invariants were considered again. An investigation was performed on whether the Object modeler identified the violated invariants. To accomplish this task, the creation of the models that were not following the invariants that were specified for the used class diagrams was attempted. The result was that the Object modeler was reliable and identified all of the tested violations.

The step provision of the data for the object attributes was evaluated by investigating whether the Object modeler considers model-specific data for the included objects correctly. Evidence was provided for some of the attributes that were included in the set of test models. Test data were chosen in such a way that it led to different results than the included default values for the attribute derivations that were included in the used class diagram. Then, the inferred attribute values were compared, to ensure that the Object modeler considered the provided evidence. The data set that is described below was used for the step evaluation of the model,

to enable the values of the included attributes to be inferred manually. These manually inferred values were compared to the values that the object modeler derived automatically.

After performing this evaluation, the result showed that the tool could be considered to have provided evidence that it produced the results that were intended for the used test cases. Instead of using default values that were specified in the class diagram, the Object modeler makes use of the user's input, when this input was included in the test cases.

To evaluate the step configuration, a sampling of two activities was performed. On the one hand, an inference that used the test cases was performed, which led to different results based on the configurations of the sampling algorithms. On the other hand, a debugger was used to follow the internal performance of the sampling algorithms. This action was performed to ensure that the Object modeler provided the implemented sampling algorithms with the correct configuration parameters, based on the user's input.

Both of the activities led to the conclusion that, for the performed tests and the used test cases, the tool considers the configuration parameters in the way that was intended by the user.

The second and last step of the evaluation of the model was considered by using a number of test cases. Following the OCL standard, atomic attribute derivations were implemented that represented the different defined OCL operations. Furthermore, mechanisms for aggregating atomic OCL derivations into more complex derivations were used. This approach was selected because it is not possible to implement every possible attribute derivation. Following the OCL standard, the possible combinations are endless. This approach resulted in a number of object diagrams of various sizes and complexities. The test cases had in common that it was possible to manually perform inferences of the included attributes in parallel with the inference that the Object modeler had the capability. The manual calculations were compared to the inferred values that the Object modeler derived.

It was realized that the tool performed inferences correctly for the considered test cases. The attribute values that the tool derived automatically corresponded to the values that were calculated using pen and paper.

Finally, the step modification of the models was evaluated. Consideration was given to whether the tool handled modifications on the provided input, i.e., the object diagrams during the performance of the inferencing. The used test cases were modified, both with regard to the structures of the models, i.e., less or more instantiated objects and relations, and with regard to the provided evidence. Then, the test cases were recalculated. Again, manual calculations were compared with the values that the Object modeler derived automatically.

The result was that the tool considered modifications of the models properly for the tested scenarios. The derived values for the included attributes matched the modifications and were in line with the manual inference that was performed.

This result concludes the present section on the requirement "the tool shall possess analysis capabilities". This section reported that the tool was successfully

used to perform analysis in several test cases. Additionally, for use in the test cases, an evaluation showed that the analysis was performed correctly, i.e., it leads to the same results as manual analysis.

9.3 The tool shall possess administrative capabilities

As described in Chapter 4, the focus of this requirement is twofold. On the one hand, the tool's capabilities for handling and presenting large-scale models are investigated, and on the other hand, support for collaborative usage of the software is considered.

In the demonstration chapter (cf. Chapter 8), usage of the tool was reported. In some cases, large models, which consist of more than 500 objects, were created. On the one hand, tool users created models of this size based on real scenarios, and on the other hand, test models were created automatically. The tool was able to handle models of this size. Table 9.4 reflects this finding.

[119] reports on the tool's analysis capabilities with regard to the handling of large-scale models. They conclude that their implementation of the cyber security modeling language CySeMoL using the presented tool "scales rather linearly with the number of modeled attack steps". Table 9.5 shows their findings.

Assets	Attack Steps	Attack Step connections	Computational time (seconds) CySeMoL	P^2CySeMoL
5	25	52	0.1	8.6
50	241	598	2.59	47.6
100	482	1203	1689.5	83.4
150	723	1808	α 45921	115.6
200	964	2413	α $3.35 * 10^6$	145.1
500	2410	6037	α $5.07 * 10^{17}$	355.2
1000	4820	11212	α $2.18 * 10^{36}$	874.7

Table 9.5: Performance of P^2CySeMoL (attack steps and attack step connections refer to P^2CySeMoCySeMoL) and α inticates estimations

To visually aid the user, the tool offers several types of support functionality (cf. Section 7.2). These include views that follow viewpoints for visualizing only subsets of the model and also the usage of templates as a means of providing an abstraction model . Furthermore, the automatic instantiation of class diagrams is supported and is also discussed as part of the fulfillment of the requirement "the tool shall support the import, editing and validation of data from external sources".

In the case of the large models that were mentioned before, the resulting object diagrams consisted of more than 20 views. For the considered cases, the tool still performed as expected. The other aspect of the requirement of administrative

capabilities is, however, not addressed in the current tool. The tool does not support web-based collaboration mechanisms, versioning systems, role management or access to the created models from mobile platforms.

9.4 The tool shall possess presentation capabilities

This requirement addresses information visualization and how insights gained from using the tool are made accessible for different audiences.

The tool supports the creation of tailored depictions of the model that are dependent on the audience. On the one hand, the author of the analysis frameworks can define viewpoints that he or she considers to be relevant (cf. Chapter 7). For example, in the case of MAP [136], the user defining the class diagram added viewpoints that represent the different included system properties. On the other hand, the user of the Object modeler has the possibility of creating user-defined views. These views do not necessarily need to be based on viewpoints. In addition, the user can create reports that describe the results of the analysis. Of particular interest for such reports are the calculation results, which can be exported, also.

An additional feature that the tool possesses to provide different stakeholders with the needed information is the impact analysis that was mentioned earlier. The decision maker can, using this functionality, identify sources for changes and specifically trace the cause for a certain analysis outcome.

9.5 The tool shall feature an extendable metamodel

This requirement addresses the Enterprise Architecture tool's capability of adapting the metamodel (called the class diagram in the tool) to cater to company-specific needs.

The presented tool addresses this requirement in several ways. The tool's component Class modeler includes an editor for creating, modifying and tailoring class diagrams. As was described in the previous chapter (cf. Chapter 8), it is possible to cover a variety of different topics, i.e., frameworks for the analysis of different system properties, which can be expressed using class diagrams. These topics are summarized in Table 9.3. By creating frameworks for the analysis of one or several properties, theoretical descriptions of those system properties are created; these descriptions correspond to a knowledge base in CAD terminology (cf. Section 4.3). It is possible to assign default values to the attributes that are included in a class diagram. These default values can be company specific. For example, to express that the employees at a certain company are more aware of cyber security aspects, because they took advanced training, the defaults can be overridden.

As was described in Chapter 7 when the distinct functionality of the tool was discussed, the object diagram can be updated, and in doing so, the user can utilize the latest version of an analysis framework. This capability goes beyond the re-

quirement that is posed in the Enterprise Architecture tool evaluations (cf. Section 7).

During the elicitation of the requirements of a tool for Enterprise Architecture analysis (cf. Chapter 4), the requirement "the tool shall support an extendable metamodel" was used even to describe the need for support for Enterprise Architecture "frameworks and standards" (cf. Table 4.2). This simplification was made based on the assumption that frameworks and standards in general and their included metamodels in particular can be captured by extended metamodels. However, the tool presented here, which is the result of the research presented in this thesis, does not have any built-in standards and frameworks. It does, however, support the manual specification for class diagrams that correspond to the metamodels found in Enterprise Architecture standards and frameworks. Additional support for the usage of standards and frameworks is not provided.

9.6 The tool shall support the import, editing and validation of data from external sources

This requirement addresses the capability of the Enterprise Architecture tool to make use of data from a third party tool. The tool is capable of doing so, in case the other tool can generate an XML-based export. This feature was illustrated in [42] and [118] . The user can define a mapping that specifies the import process, i.e., the classes that should be instantiated based on the XML file. Both a user of the Class modeler and a tool applicant of the Object modeler component can establish such a mapping. When an instantiation is performed following such a mapping, the user of the Object modeler can customize the outcome of the import. It is possible to add or remove objects and relations that connect them as well as to provide input for the values of the included attributes. Table 5 contains an evaluation that was performed by the authors of [118]. In this evaluation, a comparison between a manually created model and two automatically instantiated models is described. Considering this evaluation, it can be observed that the presented approach for the automatic creation of the architecture models has a better performance compared to the manual creation of the models.

Compared to other tools, the presented software has the weakness of supporting only XML files. Other Enterprise Architecture tools often support many of the file formats that are typically used in an office environment [166]. Some of the other available Enterprise Architecture tools also have extended validation mechanisms, which are used to identify the quality of the import data.

Table 9.6 illustrates how the tool's import capabilities can be used for faster model generation. The results were originally published in [118].

Variable	Time (hh:mm)	Entities	Relations
Modeling by [184]	05:12	20	19
Modeling using an authenticated scan	03:08	1 558	679
Modeling using an unauthenticated scan	01:23	558	462

Table 9.6: Comparison of modeling effort in hours (hh) and minutes (mm) for the proposed approach and results by Närman et al [184]

9.7 The tool shall support the storage of models, instantiating a metamodel, in a repository

This requirement addresses the Enterprise Architecture tools support for the storage of models based on metamodels.

In the tool, only local storage of the created files is supported. As was described in 9.4, the tool has been used to create models of various sizes. These models were used to evaluate the storage capabilities of the tool. An investigation was performed on whether the tool successfully saved the models and the views that were used to structure them as files. Thereafter, an evaluation was performed on whether the tool could load these files again. These tests were performed successfully for all possible models.

However, the tool has the limitation that it is not possible that two or more users create a model simultaneously or even edit one model file in parallel. This limitation makes collaborative work difficult because synchronization must be performed manually. The tool does not actively support the model evolution, i.e., it is not possible to save different versions of the models into one file. Instead, the user must actively create different models to represent different states of the model. Collaborative work is possible in such a way that several users of the Object modeler can use the same class diagram, utilizing an analysis framework in parallel. Their individual models can then be combined into a larger description using copy and paste.

9.8 The tool shall support the creation of metamodels that cover the domains of business architecture, information architecture, technology or technical architecture and solution architecture

To investigate the fulfillment of this requirement, consideration must be given to whether it is possible to create analysis frameworks that cover the enterprise con-

text, including the business architecture, information architecture, technology or technical architecture and the solution architecture.

In regard to the expressiveness of the metamodel, the tool has strengths compared to other available tools. Class diagrams, i.e., the metamodels that are used within the analysis frameworks, cannot cover only the domains that Gartner (cf. Section 4.1) requires but additionally must cover every other aspect that is relevant (for an analysis) that can be expressed in terms of a UML Class diagram [55](cf. Section 10.3). For example, the class diagram for cyber security analysis presented in [119] includes the social zone concept.

The ability to cover the business architecture, information architecture, technology or technical architecture and solution architecture was demonstrated in the analysis frameworks that were specified using the tool. In the following table (cf. Table 9.7) , the classes of some of the analysis frameworks that were specified using the tool were mapped according to the domains suggested by Gartner, to illustrate the fulfillment of this requirement [1].

Analysis framework	**Business architecture**	**Information architecture**	**Technical architecture**	**Solution architecture**
CySeMol [119]	Person	Data flow	Application server	
	Person		Application client	
	Security awareness program		Data store	
	Security awareness program		Firewall	
	Social zone		IDS sensor	
	Zone management process		Network interface	
			Network vulnerability scanner	
			Network zone	
			Operating system	
			Password account	

Continued on next page

[1] This mapping does not contain all of the classes that are included in the frameworks

Table 9.7 – continued from previous page

Analysis framework	Business architecture	Information architecture	Technical architecture	Solution architecture
			Password authentication mechanism Protocol Software product Web application Web application firewall	
The framework for interoperability analysis presented [260]	Actor	Language		Communication need
		Reference language Language translation		
The Multi-Attribute Prediction (MAP) class diagram [136]	Business process	Data set	Application component	Service requirement
	Business service Customer Organizational unit Product	Information requirement Language Representation set	Application function Application service Infrastructure function Process service interface	Stakeholder

Continued on next page

Table 9.7 – continued from previous page

Analysis framework	Business architecture	Information architecture	Technical architecture	Solution architecture
	Role		Node	
Availability Framework presented in [80]	Business process		Application component	Requirements and procurement
	Process solution of backup Operations		Application function Application service	
	Change control		Communication path Infrastructure service Node	
Utility Framework presented in [204]				Attribute requirement
				Class requirement Decision maker

Table 9.7: Ability to consider Business architecture, Information architecture, Technical architecture and Solution architecture

As described above, the class diagram can also be adapted by the user, to specifically represent the described analysis framework in the way that he or she perceives it.

9.9 The tool shall support the creation of models

The requirement "the tool shall support the creation of models" addresses the aspect of creating descriptions of enterprises. More specifically, the capability of creating an organization-wide model that covers everything from the strategy of the company down to the technical solutions (cf. Chapter 4) is addressed.

As described above, the presented tool can create models that describe almost any topic, as long as that topic can be expressed in a class diagram. In [136], an analysis framework is presented that covers the strategic dimension of organizations as well as the infrastructure and several aspects that are typically found between them. That it is possible to specify such a class diagram illustrates that the tool can fulfill the requirement that "the tool shall support the creation of models". Table 9.8 illustrates the number of models that have been created to cover either the analysis frameworks or the scenario descriptions.

Models that are created with the tool can represent organizations at a more detailed level, beyond the scope of this requirement. The resulting models not only reflect the organizations but also illustrate how the attributes of the described objects relate to one another.

Type	Number of created models
Class Diagrams specifying analysis theory	>10
Object Diagrams based on Class Diagrams capturing scenarios	>60

Table 9.8: Created models using the presented tool

9.10 General evaluation

Evaluation of the presented tool shows that the presented software tool has both weaknesses and strengths. The purpose of the research project documented in this thesis was to develop and demonstrate an Enterprise Architecture modeling and analysis tool. In this chapter, an evaluation of the tool against the requirements specified for such a tool was presented.

While evaluating the tool, it was determined that most of the users who applied the tool thus far were satisfied with its usability. Additionally, it was realized that the tool can be used for the conduct of Enterprise Architecture analyses and therefore possesses analysis capabilities. It was also determined that the tool was partly able to fulfill the requirement "the tool shall possess administrative capabilities" because it could address large-scale Enterprise Architecture models. However, the tool failed to offer a multiuser environment. During the evaluation, it was concluded that the tool possesses some presentation capabilities. This finding was realized because it was possible to fulfill the information demands of some of the stakeholders by visualizing subsets of the Enterprise Architecture models that they considered to be relevant. The evaluation also concluded that the tool allows the creation of extended metamodels that be used not only for descriptive purposes but also to conduct analysis based on those purposes. However, the tool does not feature any built-in Enterprise Architecture standards or frameworks. The tool's

ability to specify extended metamodels can be used to model the metamodels that are included in such approaches.

It was additionally realized that the tool allows for importing data from external sources, provided that these data are available in XML format. The evaluation of the requirement "the tool shall support the storage of models, instantiating a metamodel, in a repository" concluded that it is possible to save and load models that instantiate a metamodel. However, at present, the model files are stored locally and not in a data repository. The evaluation also identified that the tool allows creating metamodels that cover the domains of business architecture, information architecture, technology or technical architecture and solution architecture. Finally, the evaluation concluded that the tool could meet the requirement "the tool shall support the creation of models" because this capability helps the user to draw enterprise-wide models. In summary, the tool was able to partly or completely meet the stated requirements. Therefore, the conclusion can be drawn that the tool is a tool for Enterprise Architecture analysis.

Two requirements were clearly not met completely. The tool does not offer a multiuser environment and, in addition, does not contain a repository for storing and accessing the created models. These capabilities should, therefore, be considered to be complementary to the tool and to be part of the expected future work (cf. Chapter 12).

Chapter 10

Discussion

The result of the research described in this thesis is a tool for the analysis of Enterprise Architecture models. In the previous (cf. Chapter 9), an evaluation of this tool was presented. This evaluation used specific requirements that were elicited in the initial phase of the research project, to investigate the quality of the tool.

This research project follows a defined Design Science method that was presented in Chapter researchmethod. [33] suggests evaluating the general aspects of validity, reliability and generalizability as part of the evaluation step when following the Design Science approach. This procedure should be performed to determine the relevance and rigor of the created artifact. Again, it is important to stress the fact that the validity, reliability and generalizability of the performed research are discussed. This tool is a platform that can be used to evaluate many of the system's properties. To discuss the validity, reliability and generalizability of a specific analysis framework that describes one or several system properties is the task of the individual author and is out of the scope of the research that is documented in this thesis.

Following [33], this chapter is composed of three sections, each of which contains a discussion of one of the following qualities: validity, reliability and generalizability.

10.1 Validity

Validation is the extent to which an instrument measures what it is supposed to measure and performs as it is designed to perform [159, 187]. [14] identified four types of validity as criteria for measuring the quality, which are the following: face validity, criterion validity, construct validity and content validity. Consequently, validation is the process that determines how well the intended concept is actually being measured by an instrument.

In the context of this thesis, an evaluation of the validity of the performed work translates into a determination of whether the resulting, designed artifact is in fact

a tool for the analysis of the system's properties in terms of the organization-wide architecture models.

Even though the previous chapter (cf. Chapter 9) evaluates the outcome of the research that is documented in this thesis, there are several threats to the validity of the performed work:

Requirements

The goal of the described research project was to develop and demonstrate an Enterprise Architecture modeling tool that has a focus on system property analysis. Ideally, an already existing requirements catalog, which would preferably be provided by a third party, should have been used to identify the characteristics that such a tool should possess. However, such a catalog did not exist before this research project was conducted, and a different approach had to be selected. The tool was evaluated against a set of requirements that were derived from three different domains (cf. Chapter 4), to ensure that the goal of the research described in this thesis was achieved. First, surveys that evaluate the Enterprise Architecture tools were used because the presented tool should be comparable to other available Enterprise Architecture tools. Second, the method for the performance of Enterprise Architecture analysis was considered, to incorporate the perspective of system property analysis on the Enterprise Architecture models. Third, the architectures of the CAD tools were considered because this type of tool has a similar goal, i.e., the analysis of (architecture) models.

Only two surveys ([166] and [88]) that evaluate the Enterprise Architecture tools were used in the described research. The validity of the performed work could have been increased if a larger number of tool surveys had been considered that could have been used to identify the requirements. Additionally, only one method for the analysis of the Enterprise Architecture models was considered. The availability and consideration of several methods would have added validity to the requirements that were elicited in the domain of Enterprise Architecture analysis.

Elicitating requirements from scientific publications on CAD tools brings threats to the validity. Until now, the domain of Enterprise Architectures is not a typical area of application for CAD tools. Therefore, no CAD tool that can be used as an object of study could be identified. Moreover, CAD tools can be used for a variety of purposes (cf. Section 4.3). However, no generic CAD tool exists that can be utilized in all areas of application. Such a tool would be valuable for identifying the general requirements for a CAD tool. Instead, a consideration that is based on different areas of applications, e.g., expert systems in production planning and scheduling, had to be made (cf. Section 4.3). Deriving requirements on a CAD tool for an Enterprise Architecture from, for example, tools for production planning and scheduling, however, comes with the drawback of not establishing a perfect match. This consideration bears the risk of not identifying the relevant characteristics of a CAD tool for Enterprise Architecture or focusing on characteristics that are, in fact, less important than perceived. Moreover, the requirements were aggregated

during the elicitation process. Performing such an aggregation bears the risk of oversimplifying the mechanism that is used to evaluate the created research outcome. Such an aggregation might also have the consequence that some aspects are prioritized because of their assumed importance whereas other requirements are not considered adequately.

In summary, the considered sources that were used to identify the requirements of an Enterprise Architecture analysis tool might threaten the validity of the performed research. None of the considered domains contains an explicit, validated catalog that has the requirements for a tool for the analysis of the system properties of organization-wide architecture models that could have been used. Therefore, the threat exists that the tool does not meet all of the requirements on an Enterprise Architecture analysis tool because eventually not all of the requirements have been identified. This limitation is a threat to the fulfillment of the goal of the research that is documented in this thesis, i.e., to develop a tool for Enterprise Architecture analysis.

Evaluation

The other threat to the validity concerns the evaluation of the tool discussed in the previous chapter (cf. Chapter 9). The evaluation concluded that the presented research project succeeded and resulted in a tool for Enterprise Architecture analysis. However, there are threats to the validity of each of the eight areas that were considered when evaluating the tool. These evaluations might have been incomplete or incorrect. Next, each of the eight requirements that the tool was evaluated against in the previous chapter will be discussed with regard to the validity.

The tool shall offer a high degree of usability

In Chapter 9 the usability was evaluated. This evaluation was performed using a questionnaire that was sent to the users of the tool. The validity of this evaluation is threatened by the small sample size because only 37 users shared their experiences. A low number of samples might give too much weight to the subjective opinions. Another threat to the validity of the performed evaluation is the fact that more than one fourth (10 of 37) of the submitted answers came from (former) colleagues of the author. Additionally, 12 answers came from former students of the author. These answers might have been biased.

The tool shall possess administrative capabilities

Chapter 9 also contains a description of the tool's analysis capabilities and the proper performance of the analysis. As was described, a number of test cases were used to ensure that the tool properly infers the values of the modeled attributes. However, the number of used test cases was limited, and not all theoretically possible scenarios were covered. Instead, representative scenarios were used to ensure

that the tool and, in particular, the tool's inference engine perform analysis as expected. This procedure bears the risk that not all of the relevant scenarios were covered and that analysis in some of the cases might lead to wrong results.

The tool shall possess presentation capabilities

The tool's capabilities of addressing large models were also investigated in Chapter 9. This task was accomplished based on several tool applications that resulted in specific complex models that had many included objects and relations between them. The tool fulfilled this requirement because it could handle such large models. However, there is no upper limit for the size of an Enterprise Architecture model. Creating a model that is too large to be handled by the tool is possible. The validity of the fulfillment of this evaluation criterion is threatened if the models that were used for the evaluation are not sufficiently large.

The tool shall possess presentation capabilities

The evaluation presented in Chapter 9 continued by discussing the tool's presentation capabilities for varying groups or stakeholders. It was concluded that the tool has presentation capabilities that allow it to illustrate information for various, interested audiences. Here, the validity of the evaluation is threatened because it is not possible to predict all of the possible needs of eventual stakeholders. In addition, a subjective prioritization had to be made regarding the presentation capabilities that were offered by the tool. This arrangement was necessary due to the limited available resources, which restricted the presented project. These subjective choices also threaten the validity of the evaluation.

The tool shall support an extendable metamodel

In the evaluation chapter, it was concluded that the tool has an extendable metamodel. The term extended metamodels in the context of this thesis means not only extended to fit a specific organization but also extended to incorporate analysis theory. Using the tool, metamodels, which are class diagrams that follow the wording of the tool, can be defined and used in such a way that they are capable of describing organizations and also evaluating the systems properties of those organizations. The above conclusion was drawn because many extended frameworks were specified, with each including a metamodel that was extended by a number of authors. The validity of this evaluation is threatened because it is difficult to estimate all of the potential areas that are worthwhile considering when creating an extendable metamodel. This aspect is further discussed in the third section (cf. Section 10.3) of this chapter, which discusses the generalizability of the presented result.

The tool shall support the import, editing and validation of data from external sources

In the previous chapter, it was concluded that the tool possesses capabilities for importing, editing and validating information from external sources. As was described, this set of capabilities is achieved using files that follow the XML standard. The validity here is threatened because, thus far, only this file format has been supported. Although XML is an accepted standard that is used by many tools, it is not ensured that interesting external information is available in this file format. For some third party data sources, an import might therefore be possible.

The tool shall support the storage of models that instantiate a metamodel in a repository

The evaluation of the tool also concluded that the creation of models within the tool follows metamodels. These metamodels can be defined in the Class modeler of the tool and instantiated in the Object modeler. The evaluation additionally noted that a repository is missing. The validity of this evaluation is threatened in the sense that, given that a user is interested in a certain system property, it is not ensured that a metamodel exists that captures that specific property. Moreover, it might not be possible to use the tool's Class modeler and create a class diagram, i.e., a metamodel that describes that system property. This limitation might arise from generalization issues (cf. Section 10.3).

The tool shall support the creation of metamodels that cover the domains of business architecture, information architecture, technology or technical architecture and solution architecture

The evaluation of the tool resulted in the conclusion that this requirement was met because several authors specified analysis frameworks in terms of metamodels using the tool. These metamodels covered some or all of the 4 domains, namely, business architecture, information architecture, technology or technical architecture and solution architecture. The validity of this conclusion is threatened because these domains are not unambiguously defined. Gartner describes these areas vaguely in [89], however, but does not specify exactly the concepts that are considered to be part of each of the 4 domains. It is therefore not possible to exactly determine whether all of the concepts that Gartner regards to be included in the domains business architecture, information architecture, technology or technical architecture and solution architecture can actually be expressed by metamodels. It might be possible that someone associates a certain concept that cannot be captured in a metamodel using the tool.

The tool shall support the creation of models

Finally, in the evaluation presented in the previous chapter, it was assessed that the tool supports the creation of models. This capability was shown among others by discussing successful tool applications that led to the models. However, the conclusion might be threatened in a way that is similar to the previous requirement. It was not possible to test all of the possible models. There might be some scenarios that, using the presented tool, cannot be captured in terms of a model.

Final comments on validity

During the evaluation of the tool, investigation was performed as to whether each of the identified requirements was in fact met. It was concluded that this goal was met and that, therefore, the presented artifact is actually a tool for Enterprise Architecture analysis. As was discussed in this section, threats to the fulfillment of each of the requirements could be identified. These findings bear the consequence that the achievement of the overall goal of the research project is threatened, i.e., that no Enterprise Architecture modeling tool with a focus on system property analysis was actually developed and demonstrated.

10.2 Reliability

Investigating the reliability of a result aims at identifying the extent to which the research can be replicated. To attest that the research is reliable requires that a researcher using the same methods can obtain the same results as those of a prior research endeavor [275]. In the context of this thesis, the research at hand is the creation of a tool for the analysis of Enterprise Architecture models with regard to system properties. The question that must be answered is, therefore, the following:

> Would a researcher with the task of designing and developing a tool for Enterprise Architecture analysis obtain the same result?

Would a researcher with the task of designing and developing a tool for Enterprise Architecture analysis obtain the same result?

Typically, the initial step when performing research is to select a method to be followed. Choosing the Design Science approach to create a tool for Enterprise Architecture analysis is auspicious because it is commonly used in Enterprise Architecture research. Following a Design Science approach usually results in a requirement elicitation activity in the initial stage of the conducted project. With the given task, a consideration of common, classic Enterprise Architecture tools is used to elicit the requirements. Another area that must be considered is the analysis domain, in particular, model analysis, and even more specifically, the Enterprise

Architecture analysis domain, to ensure that the designed and developed tool meets its goal.

To elicitate the requirements while considering the existing Enterprise Architecture tools, documents that describe the requirements for those tools could be used. An alternative would be to compare the existing tools and thereby derive requirements based on identified commonalities. For the described tool, the first alternative was selected. This alternative was satisfactory. However, the second alternative could be selected, too, and would likely lead to comparable results.

To elicitate the requirements on the analysis capabilities, the available methods and approaches for Enterprise Architecture analysis and general analysis approaches that utilized architecture models are sources that could be considered.

For this research project, a subset of the available literature was considered. A threat to the reliability of the performed research is that other sources would lead to the identification of other requirements. Another threat is that an aggregation of the identified requirements was performed. A threat to the reliability is that no explicit method was followed while performing this aggregation.

However, there would likely be several common denominators between the elicited requirements on a tool for Enterprise Architecture analysis and the requirements identified in the described research work:

1. There should be a possibility to specify analysis theory for the purpose of ensuring scientifically correct and reproducible analysis of system properties
2. The attributes of modeled objects and their impact on each other should be captured to consider the enterprises as holistic systems.
3. It should be possible to consider the structure of the models
4. There should be a possibility of considering the uncertainty. This option should be possible for the analysis theory, the structure of the model, and the information used to describe a certain organization.
5. There should be a component to calculate thus far unknown values of included attributes, based on other known attribute values, to evaluate the system properties of Enterprise Architectures.

In the discussed research project, these requirements raised the need to make decisions in the following areas (cf. Chapter 5.4):

- Overall tool architecture
- Platform
- Modeling language
- Inference engine
- Level of abstraction
- Cyber security modeling

These areas were useful for providing guidance during the design of the tool. However, the reliability of the performed research is threatened. It is not guaranteed that these are all of the relevant areas for decision making.

There are additional threats to the reliability that concern the decisions made for each of the identified areas. These threats will be discussed next.

Design option I: Overall tool architecture

There are many possible alternatives for the design of a complex application such as the tool that is the result of the research activity described in this thesis. Often, it is difficult to compare different architectures [253].

However, deciding on the architecture of the software is fundamental and has a strong impact on the characteristics of the resulting tool. Furthermore, the overall architecture is generally fixed and cannot be modified or even replaced, which is the case for features that are built on top of a specific architecture.

In the context of the performed research, a decision regarding the software architecture must be made with regard to the context of the performed research, its goal and, in particular, the requirements posed against the tool that should be the outcome of the research endeavor.

While deciding on the tool's architecture, the aspect of having two separated user groups who are applying the tool must be considered (cf. Section 7.3). Another relevant factor is the project dynamics. These needs create the requirement for a flexible architecture that allows the fast and uncomplicated addition or modification of features to the tool.

For the tool, an architecture that is composed of two components was selected. Each user group is supported with one of these two components. The components share a core, where common functionality is implemented. Specific functionality that supports one specific user group is offered solely by the dedicated component. Using this architecture satisfied the research project's needs.

The reliability of the performed research is threatened here because the architecture of a tool for the analysis of Enterprise Architecture models could be designed in many different ways. The presented architecture was derived based on experience from previous prototypical implementations of the tool, and avoiding redundant code was prioritized. However, there might be alternative architectures that outclass the used design. Especially, the focus on redundancy reduction threatens the reliability of the project. There might be other aspects that are of greater importance. Prioritizing those aspects might result in a different overall tool architecture.

Design option II: Platform

There are several rich client development platforms that are available that could have been used to realize a tool for Enterprise Architecture analysis. The Eclipse Rich Client platform was selected due to the platform's powerful features and lively and supportive community. During the performance of the research project, this decision had a positive impact on the outcome because the platform turned out to provide useful components. Additionally, bugs that were found were often fixed within a short period of time.

The reliability of the performed research is threatened because the powerfulness and the supportiveness of a platform are criteria that are difficult to quantify. There are other platforms, especially when not limiting oneself to the consideration of Java-based platforms. One of those platforms could be chosen, also. Eventually, these platforms possess more suitable features or an even more active community. The fact that it is theoretically possible to implement a platform from scratch is also a threat.

Design option III: Modeling language

Enterprise Architecture is an approach that makes heavy use of metamodels, providing a general modeling language and models that instantiate those metamodels, to describe a specific scenario. In the tool that was the outcome of the research described in this thesis, extended metamodels are used that describe analysis theory for system property analysis. These extended metamodels are specified as UML Class diagrams and are instantiated as models in terms of UML Object diagrams.

UML is most likely the most common modeling language to be used within a variety of domains. However, it is by far not the only possible language that could be used for Enterprise Architecture analysis. There are a number of Enterprise Architecture modeling languages, and the number of general purpose modeling languages is even larger. It is, therefore, difficult to obtain a complete overview and even more difficult to compare all of the possible candidates to each other. Identifying the best modeling language for the presented tool is difficult if not impossible.

The reliability of the performed research is threatened because alternatives other than UML might be selected. UML was selected because it is frequently used and offers a balance between the usability and expressive power. However, there might be modeling languages that offer an even better balance. It also might be possible that there are languages that have the same usability and expressive power and that possess additional features that are useful within a tool for Enterprise Architecture analysis compared to UML. Such languages were not found during the design of the tool. A researcher who has the task of redesigning and redeveloping an Enterprise Architecture analysis tool might, however, be able to identify such a language.

Design option IV: Inference engine

The tool has a built-in inference engine that utilizes the Predictive, Probabilistic Architecture Modeling Framework (P2AMF) (cf. Section 5.4) to derive unknown attribute values. This engine processes input that is specified in an extended version of the Object Constraint Language OCL. This arrangement allows the consideration of dependencies between the attributes of the modeled objects as well as structural aspects of the model. Moreover, because OCL was extended to support probabilistic reasoning, uncertainty, in particular definitional uncertainty, theoretical

heterogeneity, causal uncertainty, empirical uncertainty and structural uncertainty can be considered during the performance of the inferencing.

Regarding the inference engine, the reliability of the performed research is threatened for several reasons. It might not be sufficient to evaluate the attributes based on other attributes and the structure. There might be other, additional characteristics of the model that are relevant to consider during the inferencing performance. Additionally, there might be aspects that are relevant for Enterprise Architecture analysis that cannot be captured using OCL. This possibility might, for example, be due to unsupported data types, insufficient expressiveness or imprecision that might be identified.

Uncertainty might also not be addressed sufficiently, which is another threat to the reliability. Eventually, it is not sufficient to consider definitional uncertainty, theoretical heterogeneity, causal uncertainty, empirical uncertainty and structural uncertainty while performing Enterprise Architecture analysis.

Even though the inference engine might consider the relevant aspects of the models that ought to be evaluated, i.e., the dependencies between the attributes and structure of the model, the relevant aspects with regard to uncertainty in the reliability is still threatened. There might be alternative ways of realizing the inference engine. Instead of an extended version of OCL, other languages, which allow querying models and derive unknown attribute values, might be used. During the performed research project, a better language was not identified compared with the chosen language. However, the reliability is threatened because such a language might exist. Such a language might possess advantages over the selected, extended OCL, such as allowing faster inferences or a more user-friendly specification of the rules to infer attribute values.

Finally, the selection of the supported inference algorithms (cf. Section 5.4): forward sampling, rejection sampling and Metropolis Hastings sampling is also a threat to the reliability of the performed research. There are a number of alternative sampling algorithms, and it might be the case that one or several other sampling algorithms could be selected.

Design option V: Level of abstraction

A balance must be found between high expressive power accompanied by large and complex models and simplified scenario descriptions that have more easily understood models. The field of Enterprise Architecture especially aims at finding a good compromise, to cover organizations on a holistic level and still provide valuable information for interested stakeholders. This challenge was relevant for the presented tool. A tool for Enterprise Architecture analysis must address the question of how complex models can be presented in such a way that the contained model is easy to grasp. The tool supports the use of templates as a means of reducing the visual complexity.

The reliability of the performed research is threatened because the selected level of abstraction was chosen without any scientific foundation. Instead, the usage of

templates was identified to be convenient and supportive for the creation of large models. However, because these are subjective criteria, they threaten the reliability of the project. For the tool presented in this thesis, the decision was made based on the request by the tool users. Other users might have other opinions.

It might be identified that there is no need for the templates at all. The reliability is furthermore threatened because the offered templates might be on the wrong level of abstraction. It might be the case that it is necessary to visually aggregate the models even more, to ensure understandable models. On the other hand, it might be the case that the supported templates are abstracting too much. Templates might visually concentrate the models to such an extent that the depictions are less usable. This process could lead to a decreased usability of the tool.

Another design decision made for the tool presented was to reduce the visual complexity and thereby the tool's usability by allowing the definition of the viewpoints and creating views that confirm these viewpoints. This decision is another threat to the reliability. It might be the case that the viewpoints and views do not increase the usability and instead cause confusion. The decision to support viewpoints and views was made because these concepts are commonly used within the field of Enterprise Architecture. However, the reliability of the presented work is threatened because it was not scientifically evaluated if the support for the viewpoints and views is actually appreciated by the users of the tool.

Design option VI: Cyber security modeling

The need to explicitly support cyber security modeling arose because the research documented in this thesis was partly conducted in the context of a larger research project that aimed to provide decision support with regard to the design of industrial control systems from a cyber security perspective (cf. Chapter 1). One of the outcomes of this cyber security research project was CySeMoL, the Cyber Security Modeling Language and its successor, P2CySeMoL, the Predictive, Probabilistic Cyber Security Modeling Language. Parallel to the development of this language, an investigation was performed on how the tool that is presented in this thesis could be designed and extended to provide support for CySeMoL.

The decision was made to support CySeMoL by providing a means for attack graphs. This decision, however, is a threat to the reliability. As was described in Section 5.6, there are other approaches that can be used for organization-wide security modeling. It might be the case that attack graphs are not the best way for accomplishing this goal. Compared to the functionality that is currently available, other modeling techniques have, for example, better usability, more insightful analyses or consideration of organizations on a level of detail that has thus far not been possible but could exist.

Final comments on reliability

The final component of the tool that must be discussed is the design of the user interface. The layout, design and structure of the menus, the canvas holding the model and other components that allow the user to interact with the tool can be realized in a variety of ways. The user interface of the presented tool was strongly inspired by other modeling tools, specifically other modeling tools that were built on the same platform. A threat to the reliability is that there is no standard for the design of the user interfaces. In particular, there is no document that defines the explicit requirements on the user interface of the Enterprise Architecture tools. A researcher who uses the same methods might therefore develop a different design and layout for the user interface.

In summary, it can be realized that many threats to the reliability of the presented research exist. The elicited requirements on a tool for Enterprise Architecture analysis are threatened as well as the areas for the design decisions. Even for all of the areas of decision making that were considered during the design of the tool, there are many threats. Finally, even the reliability of the creation of the user interface is threatened.

10.3 Generalizability

Discussing the generalizability of a scientific contribution aims at investigating whether the research results can be extrapolated to the larger population [252, 275].

In the context of the research that is described in this thesis, evaluating the generalizability translates into identifying other domains in which the presented tool can be applied. As was described, that tool was successfully used to evaluate more than 10 different system properties (cf. Chapter 8). In additional frameworks that were used to evaluate other system properties, such as the properties identified in [269] , [93] or [222] could be realized, also. This arrangement can be accomplished if the system properties can be expressed in terms of class diagrams and instantiated in terms of object diagrams to evaluate a specific scenario. It was also reported that the tool allows describing technical systems such as computer networks and social systems that describe the organizational structure of the enterprises (cf. Section 9.8). In general, all of the aspects that can be perceived as objects and that can be described with characterizing attributes can be captured by the tool.

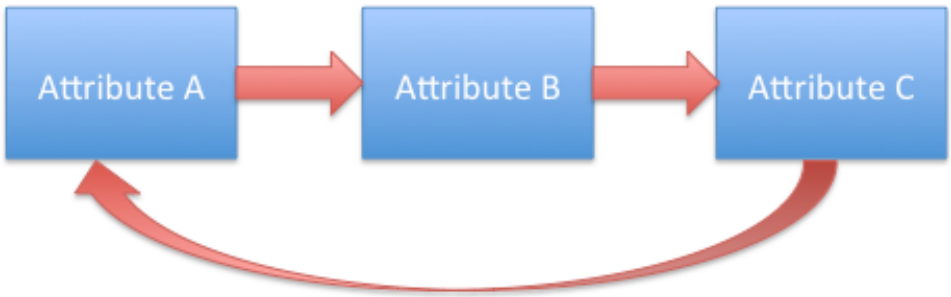
The present tool does not allow for analysis considering the temporal aspects [1] . In general, it cannot describe that the value of attribute Y at t=1 depends on the value of attribute X at t= 0. In [180] and [184] , a framework for the analysis of the

[1] It is important to understand that temporal analysis is not the same as sampling according to Metropolis-Hastings, which uses a Markov Chain. The tool is able to perform the latter; however, it fails on the former. Using a Markov Chain adds a time perspective, which however is global, i.e., from sample to sample. For one specific sample, all of the attributes are in the same time step, whereas the temporal analysis would require that even within one sample, different time steps could be considered.

data accuracy is presented. The authors of this framework attempted to overcome the mentioned weakness by describing several points in time within the same model. These points in time are related using a predecessor-successor relation. In this way, consideration over time is made possible. However, this solution is not dynamic compared to simulations in which a model updates itself incrementally. Instead, the framework that is presented in [180] and [184] considers all of the time steps at once. The described limitation restricts the applicability of the tool to static areas in which there is no need to consider any of the system dynamics.

Another aspect that the tool cannot handle is circular relationships among attributes of the created object diagram. It is not possible to express that attribute A impacts attribute B, which in turn impacts attribute C, and C then has an impact on A (cf. Figure 10.1). Instead, the tool expects the object diagrams to have a directed graph structure for the attributes. Class diagrams do not necessarily need to have this characteristic. It is important to ensure, using multiplicities and invariants, that their instantiating object diagrams fulfill this criterion.

Figure 10.1: The not supported circular relationships between attributes



Chapter 11

Information of relevant audiences

This chapter describes the communication step of the used method. As described in Chapter 2, the final step is to inform the relevant audiences about the problem and its importance, the artifact, its utility and novelty, the rigor of its design, and its effectiveness.

11.1 Information of relevant audiences

The relevant audiences can be divided into two groups. On the one hand, this group consists of the community of scientists who have similar research interests. Research groups that are developing tools for the analysis of Enterprise Architecture models naturally have the highest interest because their topic is fairly close. Groups that perform research in the field of Enterprise Architecture analysis in general can also fairly easily relate to the presented tool. Further academics who are investigating other aspects of Enterprise Architecture or architecture analysis for other domains are relevant audiences, also.

The remaining relevant group to be informed is the tool users who actually apply the tool presented in this thesis. This group should not only take notice of the presented artifact but also start using it, after having identified its utility.

11.2 Presentation of the tool for academic audiences

This section describes how the tool has been presented to scientific audiences.

[135] is the first presentation of an idea for the tool-based analysis of Enterprise Architecture models. This article presents the software on a theoretical level, without discussing any implementation details. The first publication that illustrates the architecture of the tool is [64], which also describes how the tool can be used for the analysis of maintainability. A more detailed presentation of the underlying formalisms and technical aspects was then published in [40] . In [44] the implementation of the PRM, a formalism for supporting Enterprise Architecture analysis was

illustrated for the first time. [259] sketched the idea of using the P2AMF for the first time, referring to it as pi-OCL.

The tool's capabilities for automatically creating architecture models were presented in [42] . [43] builds on top of [259] and actually illustrates the P2AMF component of the tool. Finally [119] demonstrates the usage of templates.

11.3 Presentation of the tool for tool users

As was described in the demonstration of usability chapter, the tool was presented to possible tool users who were interested in either the specification of analysis frameworks or the usage of such frameworks to analyze Enterprise Architectures. In total, more than 15 theory experts used the tool to specify more than 10 analysis frameworks [246, 119, 186, 182, 260, 205, 184, 183, 263, 134, 204, 180, 136] . On the other hand, almost 50 users applied the analysis component. These users were either students (29) or practitioners (20).

Chapter 12

Future Work

This chapter describes topics for future work that were identified during the performance of the research work described in this thesis. These topics were, on the one hand, identified during the evaluation of the tool, as was described in the evaluation chapter (cf. Chapter 9). On the other hand, topics were identified during the demonstration phase (cf. Chapter 8). More specifically, the theory experts who used the tool for the specification of analysis frameworks had ideas that have not yet been implemented. Future research that regards the presented tool can be separated into four categories. The first category is further development based on the evaluation of the tool (cf. Chapter 9). The second category is future work within the tool's focus, i.e., providing decision support using the Enterprise Architecture analysis. The third category arises because the tool was developed in the context of a research project that aims to support the analysis of enterprise-wide cyber security analysis; this domain should be considered specifically. For the fourth category, an investigation could be perform on whether the functionality that is offered by other Enterprise Architecture tools should be added to the presented implementation.

12.1 Future work based on the evaluation of the presented tool

While evaluating the tool (cf. Chapter 9) against the previously elicited requirements, it was realized that the tool failed to completely meet the two requirements. It was identified that the tool neither offers a multiuser environment nor contains a repository for storing and accessing the created models. Complementing the tool based on this realization, as part of the future work, is therefore worthwhile considering.

For both of the identified weaknesses, generic solutions are available that could be integrated into the tool. SVN [178], CVS [50], Maven [74] and Mercurial [175] are well-proven approaches for version management that are successfully used in large-scale software development projects such as the development of the Linux

Kernel [160]. The used rich client platform (Eclipse Rich Client Platform) offers components that can be used to realize integration with these version management solutions. Any current operating system features multiuser management [218], and there is plenty of literature available, provided by operating system vendors, which describes these aspects [172, 189, 198]. By more deeply integrating the tool with the operating systems, multiuser management could be achieved.

12.2 Future work with regards to Enterprise Architecture analysis

Pertaining to the analysis of architecture models, potential domains of extensions include results visualization, modeling techniques, analysis techniques, and automated data collection. The tool, in its current implementation, has some ability to visualize the results of the architecture analysis. However, future work might investigate how the results can be depicted when adjusted for different involved stakeholders and how the understanding of the analysis results can be eased.

Concerning the area of the modeling techniques for the analysis of certain system properties, some of the tailored visualizations are common. An example is the attack graphs [145] that the tool already supports. These are a common means in the field of cyber security analysis to visualize sequences of attacks. In the future, it might be considered to support such specific ways to model information for other system properties, also.

Another aspect in the area of modeling is support for a conditional existence [177]. If the tool and the used inference engine would allow the user to describe that certain modeled entities exist only when other systems are in place, the analysis process could be sped up and could deliver more realistic results. For example, in the case of cyber security analysis, it would be meaningful to capture that a firewall exists only if it is provided by an operating system.

In the field of modeling, it would in addition be helpful if the tool's capabilities for specifying P2AMF-based analysis frameworks would be improved. Currently, the tool provides capabilities for specifying P2AMF equations; however, it is easy to lose the big picture. Information is spread over many dialogs and, in particular, the reuse of code is difficult. Because the tool is built on top of the Eclipse Rich Client Platform, the Eclipse Integrated Development Environment would be a natural source of inspiration for generating a simplified and unified environment for the specification of P2AMF-based analysis frameworks.

Finally, the modeling of alternatives based on a given scenario could be fostered. The tool could, for example, provide suggestions for variations that are based on the used class diagram. Similar to software development, it could be interesting to support branching and eventually the future merging of models. This capability would enable the user to focus on a certain aspect of the model. He or she could investigate this aspect and, when obtaining satisfactory analysis results, merge it into the main model branch.

In the field of analysis techniques, several areas can be considered to be a part of future research. The capabilities of the inference engine can be extended to help the tool users draw better conclusions and obtain better decision support. The possibility of performing sensitivity analysis [154] would allow the user to identify the strength of impact that certain attributes have on a specific outcome. This arrangement would help to identify the most rewarding decision.

The inference algorithms that are provided by the inference engine could also be regarded with respect to their potential for optimization or redesign. It might be worthwhile considering alternatives to the inference algorithms that are already in place. Currently, the sampling process often requires a large number of samples, which results in a significant duration, to deliver accurate calculation results. It would be possible to investigate whether the inference algorithms could be parallelized and, in the specific case of the Metropolis-Hastings algorithm, might be worthwhile evaluating how the first good sample could be identified in a faster way. Moreover, a scenario comparison is an interesting analysis feature that could be offered by the tool. This approach could be insightful when comparing object diagrams with regard to their structure. Here, the tool could help to visually identify the differences among several models. Even more interesting might be to compare the values of the included attributes after an inference has been performed. The tool could offer a tabular comparison among the evaluation results of the considered scenarios. In such a comparison, the tool could help to identify significant differences and thereby potentially interesting scenarios. On the other hand, irrelevant or unsatisfactory alternatives could also be found.

Finally, in the area of analysis techniques, simulation of the model and automatic identification of the best solution is an interesting topic for future research. With the help of utility functions [204], the tool could evaluate the usefulness of an object diagram according to the user's perception. Once one initial object diagram was created, the tool could automatically generate alternatives by varying the included objects and attribute values. The generation of the alternatives could be constrained by the tool's user to ensure that only plausible scenarios would be considered. Evaluating the utility of all of the generated reasonable alternatives would allow the tool to find the best solution(s).

In addition, work on the tool's capability of automatically generating models based on data from external sources has been performed (e.g., using a vulnerability scanner, as demonstrated in [42]). This area might be further investigated as well. In [70] a process for the automatic generation of Enterprise Architecture models was outlined. This approach could be considered to support this process or a similar process. Other potential sources include ERP systems, as was proven in [41] and [106]. Furthermore, [71] mentions configuration management databases, project portfolio management tools, enterprise service buses, change management tools and license management tools as possible contributors. Additionally, access control lists and UDDI registries contain information that might be relevant in many Enterprise Architecture models. How information about business processes and organizational structure can be gathered might be considered also because most

of the data sources that have been mentioned are on a technical level.

12.3 Future work supporting cyber security analysis

To support the performance of cyber security analysis in general and using CySeMol (cf. Section 8.2) in particular, several areas for future work were identified.

To simplify the data collection, consideration could be given to in-depth integration of vulnerability scanners and other software tools that assess computers, computer systems, networks or applications. This approach should be invested in if, based on the results of these tools, models for cyber security analysis can be created completely automatically. This finding would be interesting from both a cost and data quality perspective.

The modeling language CySeMol could also be directly integrated into the tool. Specific wizards, tutorials and helping functionality that describe the theoretical foundations of CySeMol to ease its application and make the tool usage more intuitive could be added.

Currently, the tool supports the generation of attack graphs. In the future, consideration could be given to supporting other cyber security modeling notations, also.

12.4 Enhancement of the tool inspired by other Enterprise Architecture tools

Topics for the third category of future work, the enhancement of the tool in areas other than decision support and architecture analysis, can be identified based on the Enterprise Architecture tool surveys that were considered as parts of the requirements elicitation process (cf. Chapter 4).

One area of improvement is usability. This area includes the eased model creation, manuals, tutorials and guides, online communities and white papers that describe successful applications. These aspects are already covered to a certain extent; however, for commercial tools, these topics must be continuously considered to attract new audiences.

Making the tool available for platforms other than MAC OS X and Windows, which are currently supported, can also foster usability. Mobile platforms such as IOS, Windows Mobile and Android could be considered, as was proven in [20] . On the other hand, Linux might also be an option.

In the evaluation of the tool (cf. Chapter 9), it was discussed that the current tool versions possess some weaknesses with regard to their repository functionality. For now, only very limited capabilities for version handling are available. In a future tool, version support for different historical states of the created models might be added. Using a repository to store the models could also support the collaborative work of physically separated teams. Especially because Enterprise Architecture is an interdisciplinary field, it would be meaningful to provide a tool-based solution for

different departments or domain specialists to contribute to one holistic Enterprise Architecture model. Additionally, user management that includes different roles could be of interest. This construct would allow dividing the tool users into different categories. For example, only selected users could have privileges to edit the models, and others could only conduct analysis; a third group could only consider the models.

From the perspective of creating presentations, there are several more advanced visualization techniques that could be employed in the tool. The field of software cartography [146, 152] aims to support decision making on an Enterprise Architecture level using visual representations that depict only the information that is currently needed. The authors of [37] describe how software cartography can be used in an Enterprise Architecture tool. The authors create visualizations using model transformations, an approach that could be utilized in the tool also, specifically because the implementation presented in [37] also uses components of the Eclipse Modeling Framework (cf. Section 7.3).

Another aspect that is worthwhile considering in the future is the interaction with other tools. On the one hand, the import of existing information can be improved. The use of models created in other Enterprise Architecture tools would be helpful and would speed up the model creation process. Additionally, the reuse of information that is available in the other tools contributes to having faster generation of the models. Relevant sources can, for example, be modeling tools for specific aspects, such as business modeling tools or software with a focus on the modeling of network topologies. Supporting the import of data that is available in spreadsheets, presentations or even text documents could be helpful capabilities to develop. One critical task for the tool, which a future implementation should address, is to check for redundant or conflicting information, to ensure a high level of data quality. However, topics of interest are not limited to the import of information; also, the export of both the models and the calculation results should be considered in the future. It would be helpful if the tool could create presentations based on views that the tool user creates.

Chapter 13

Conclusions

The purpose of the research described in this thesis was **to develop and demonstrate an Enterprise Architecture modeling and analysis tool**. Fulfilling this goal included the following subgoals (cf. Section 1.2):

1. Eliciting requirements on a tool for Enterprise Architecture analysis
2. Designing and developing a tool that considers these requirements
3. Evaluating the quality of the created tool with regard to the identified requirements
4. Demonstrating this tool for both academics and practitioners

The research presented in this thesis was performed following the Design Science approach. In Chapter 4, the requirements for a tool for Enterprise Architecture analysis are presented as a fulfillment of subgoal 1. These requirements were derived while considering the evaluation criteria for the evaluation of Enterprise Architecture tools and while studying a method for Enterprise Architecture analysis as well as regarding CAD tools and expert systems.

Chapter 5 describes design decisions that were made to fulfill these requirements. These decisions concerned the overall tool architecture, the used development platform, the modeling language, the offered inference engine, and the level of abstraction. Additionally, a decision regarding the support for cyber security was necessary because the presented tool was partly developed in the context of a research project that aims to evaluate industrial control systems from a cyber security perspective. Chapter 6 illustrates the development process that is used to develop the tool for Enterprise Architecture analysis, and Chapter 7 discusses the resulting artifact and its software architecture. In combination, Chapter 5, 6 and 7 describe the fulfillment of subgoal 2.

Chapter 9 and 10 discuss how the presented research meets subgoal 3. Chapter 9 describes the evaluation of the tool against the requirements described in Chapter 54. Chapter 10 discusses the validity, reliability and generalizability of the performed research.

Finally, Chapter 8 and Chapter 11 describe the fulfillment of subgoal 4. In Chapter 8, the practical usage of the tool is discussed. More than 10 analysis frameworks were specified using the tool. Additionally, more than 50 users conducted architecture analysis using the tool. Chapter 11 elaborates on the presentation of the performed research in terms of scientific publications.

The tool for Enterprise Architecture analysis was one of the outcomes of a larger research project that aims for decision support with regard to the design of industrial control systems from a cyber security perspective. However, even though the tool was partly developed in this project setup, this tool is not limited to the analysis of cyber security aspects and is also not limited to the analysis of industrial control systems. Instead, the tool can be used to analyze a broad range of system properties on an enterprise-wide level (cf. Section 10.3).

The result of the research described in this thesis is the development of an Enterprise Architecture tool that has an inverse focus compared to most of the other available software products that are available to support Enterprise Architecture endeavors. Many of the available Enterprise Architecture tools have a strong focus on the modeling and documentation aspects [166]. Within those tools, the performance of the analysis is more a bonus feature than a part of the core uses that are supported. The users are given comparably less powerful analysis functionality. On the other hand, the creation of models is fairly easy. Those tools often feature several ways of creating architecture visualizations and documenting the state of the enterprise.

The tool that is the outcome of the research presented in this thesis has comparably weak modeling capabilities, if the model should be used only as a means of documentation. Instead, in the presented tool, models cater to the input of the analysis. The tool does not consider analysis to be something that a user might eventually perform. Instead, it assumes that a tool user already, from the start, intends to conduct an evaluation of the described Enterprise Architecture. This focus is reflected in the tool architecture, specifically in its separation into two components, the features that the tool offers and the recommended workflow. Instead of using a standardized metamodel, which eventually might be slightly adapted to fit a company's needs, the tool is built on the idea of defining the analysis frameworks by extended metamodels that incorporate analysis theory. This approach results in two components: one component for the specification of extended metamodels, which are class diagrams that follow the UML nomenclature, and another component for the application of the analysis frameworks and the usage of the included metamodels. The tool features a powerful inference engine, which is to be used to derive unknown attribute values. Moreover, several other features are available that either simplify the specification of the analysis frameworks or simplify their application (cf. Chapter 7). Compared to other tools, the tool developed here is not tailored to support one or several selected fixed system properties. Instead, it provides interested audiences with a platform that can be used to describe and analyze a variety of different system properties (cf. Section 10.3).

The author, as described in the process section, performed multiple roles in the described research project. He elicited requirements and designed the architecture of the Enterprise Architecture analysis tool. Following the communication step of the applied method, relevant audiences were informed about the tool and the development process. In this way, the author acted as an interface between the groups that were interested in the usage of the tool and the development team. As part of the requirements elicitation and architecture derivation step, he contributed to many research projects that led to the implementation of multiple tool features. This approach results in the fact that the author is the main author of some features that can be found implemented in the tool. For other features, he had only a supporting role or primarily worked as a mediator between the developers and users who demanded new features or functionality.

The questionnaire used for the usability evaluation (cf. section 9.1)

Evaluation of KTH's Enterprise Architecture Analysis Tool

Please contribute to the improvement of KTH's Enterprise Architecture Analysis Tool and my Ph.D. thesis in particular. I would appreciate if you could share your experience regarding the usability of our tool.
The questionnaire only consists of 7 questions and you will be done within less than 5 minutes.

Your answers will be kept confidential.
In case of questions please contact me at markusb@ics.kth.se
Thank you for your contribution

* Required

0. **My goals for the Enterprise Architecture Analysis Tool were met** *
   Did the tool provide you with what you needed to achieve your goal? Did the tool offer you support for analysis of a scenario when you wanted to do so? Did the tool support you when you wanted to specify an analysis framework?
   Mark only one oval.

   - ◯ strongly agree
   - ◯ agree
   - ◯ neutral
   - ◯ disagree
   - ◯ strongly disagree
   - ◯ no opinion

0. **I received the results I expected from the Enterprise Architecture Analysis Tool** *
   Did your tool application generate analysis results if you used the tool for analysis? Did you create an analysis framework if that was your goal?
   Mark only one oval.

   - ◯ strongly agree
   - ◯ agree
   - ◯ neutral
   - ◯ disagree
   - ◯ strongly disagree
   - ◯ no opinion

0. **I completed the task within the Enterprise Architecture Analysis Tool quicker compared to using other tools that I have at my disposal** *
   Did you get results faster using the Enterprise Architecture Analysis Tool compared to if you would have used other available software products.
   Mark only one oval.

   - ◯ strongly agree
   - ◯ agree
   - ◯ neutral
   - ◯ disagree
   - ◯ strongly disagree
   - ◯ no opinion

0. **I had a pleasant experience with the Enterprise Architecture Analysis Tool when using it** *
   Did the tool appeal to you?
   Mark only one oval.

   - ◯ strongly agree
   - ◯ agree
   - ◯ neutral
   - ◯ disagree
   - ◯ strongly disagree
   - ◯ no opinion

0. **The user interface of the Enterprise Architecture Analysis Tool helped me avoid making errors** *

Mark only one oval.

- ◯ strongly agree
- ◯ agree
- ◯ neutral
- ◯ disagree
- ◯ strongly disagree
- ◯ no opinion

0. **I was able to recover when I made an error** *

Were you able to correct mistakes that you made?

Mark only one oval.

- ◯ strongly agree
- ◯ agree
- ◯ neutral
- ◯ disagree
- ◯ strongly disagree
- ◯ no opinion

0. **The user interface of the Enterprise Architecture Analysis Tool was predictable** *

Mark only one oval.

- ◯ strongly agree
- ◯ agree
- ◯ neutral
- ◯ disagree
- ◯ strongly disagree
- ◯ no opinion

0. **When did you use KTH's Enterprise Architecture Analysis Tool the last time?** *

Mark only one oval.

- ◯ 2014
- ◯ 2013
- ◯ 2012
- ◯ 2011

Bibliography

[1] Pekka Abrahamsson, Outi Salo, Jussi Ronkainen, and Juhani Warsta. 2002. Agile software development methods: Review and analysis. 6.1

[2] Software AG. 2014. ARIS architect & designer. URL `http://www.softwareag.com/corporate/products/aris/bpa/products/architect_design/overview/default.asp`. Accessed: May 2014. 3.2

[3] Software AG. 2014. ARIS business process analysis platform. URL `http://www.softwareag.com/corporate/products/aris/bpa/overview/default.asp`. Accessed: May 2014. 1.1

[4] Stephan Aier, Sabine Buckl, Ulrik Franke, Bettina Gleichauf, Pontus Johnson, Per Närman, Christian M Schweda, and Johan Ullberg. 2009. A survival analysis of application life spans based on enterprise architecture models. In *EMISA*, pages 141–154. 3.3

[5] David H Akehurst and Behzad Bordbar. 2001. On querying uml data models with ocl. In *UML 2001 The Unified Modeling Language. Modeling Languages, Concepts, and Tools*, pages 91–103. Springer. 5.3, 5.3, 7.3

[6] alfabet. 2014. alfabet planningIT. URL `http://www.alfabet.com/en/it-planning-reality/`. Accessed: May 2014. 3.2

[7] Agile Alliance. 2001. Agile manifesto. *Online at http://www. agilemanifesto. org*. 6.1, 6.1

[8] OSGi Alliance. 2014. OSGi release 5. URL `http://www.osgi.org/Specifications/HomePage`. Accessed: May 2014. 7.3

[9] David J Anderson. 2004. Feature-driven development. *Microsoft Corporation, October*. 6.1

[10] Niklas Arnel, Kristina Ernsell, Christian Wemstad, and Hobrik Wärnegård. 2013. Approaches to calculate it costs at amf using models - student report. Technical report, Kungliga Tekniska Högskolan. 3.3

[11] Michael Lyle Artz. 2002. *Netspa: A network security planning architecture.* PhD thesis, Massachusetts Institute of Technology. 1.1

[12] avolution. 2012. The abacus enterprise architecture methodology - 9 steps to architecture success. 3.2

[13] avolution. 2012. Ea management (eam) suite. URL `http://avolution.com.au/resources/resource-center/resource-files?dfID=239`. Accessed: May 2014. 3.2

[14] Earl R Babbie. 2013. *The practice of social research.* Cengage Learning. 10.1

[15] Pavel Balabko and Alain Wegmann. 2006. Systemic classification of concern-based design methods in the context of enterprise architecture. *Information Systems Frontiers*, 8(2):115–131. 3.2

[16] Joseph Barjis. 2007. Automatic business process analysis and simulation based on demo. *Enterprise Information Systems*, 1(4):365–381. 5.3

[17] David Garduno Barrera and Michel Diaz. 2013. *Communicating Systems with UML 2: Modeling and Analysis of Network Protocols.* John Wiley & Sons. 5.3, 5.5

[18] Len Bass, Paul Clements, and Rick Kazman. 2003. *Software architecture in practice.* Addison-Wesley Professional. 3.3

[19] Jeremy Bentham. 2007. *An introduction to the principles of morals and legislation.* Read Books. 3.3

[20] Maxime Bernaert, Joeri Maes, and Geert Poels. 2013. An android tablet tool for enterprise architecture modeling in small and medium-sized enterprises. In *The Practice of Enterprise Modeling*, pages 145–160. Springer. 12.4

[21] A-J Berre, Brian Elvesæter, Nicolas Figay, Claudia Guglielmina, Svein G Johnsen, Dag Karlsen, Thomas Knothe, and Sonia Lippe. 2007. The athena interoperability framework. In *Enterprise Interoperability II*, pages 569–580. Springer. 1.1

[22] Graham Berrisford and Marc Lankhorst. 2009. Using archimate with togaf. *Retrieved October*, 20:2010. 5.3

[23] Anandhi Bharadwaj, Mark Keil, and Magnus Mähring. 2009. Effects of information technology failures on the market value of firms. *The Journal of Strategic Information Systems*, 18(2):66–79. 3.3

[24] Enrico Biermann, Karsten Ehrig, Christian Köhler, Günter Kuhns, Gabriele Taentzer, and Eduard Weiss. 2006. Graphical definition of in-place transformations in the eclipse modeling framework. In *Model Driven Engineering Languages and Systems*, pages 425–439. Springer. 7.3

[25] Enrico Biermann, Karsten Ehrig, Christian Köhler, Günter Kuhns, Gabriele Taentzer, and Eduard Weiss. 2007. Emf model refactoring based on graph transformation concepts. *Electronic Communications of the EASST*, 3. 7.3

[26] Stefano Bistarelli, Fabio Fioravanti, and Pamela Peretti. 2006. Defense trees for economic evaluation of security investments. In *Availability, Reliability and Security, 2006. ARES 2006. The First International Conference on*, pages 8–pp. IEEE. 5.6

[27] BiZZdesign. 2014. BiZZdesign architect. URL `http://www.bizzdesign.com/tools/bizzdesign-architect/`. Accessed: May 2014. 3.2

[28] Grady Booch. 2006. *Object Oriented Analysis & Design with Application*. Pearson Education. 5.3

[29] Grady Booch, Ivar Jacobson, and Jim Rumbaugh. 2000. Omg unified modeling language specification. *Object Management Group ed: Object Management Group*, page 1034. 1.2

[30] Tim Boudreau, Jaroslav Tulach, and Geertjan Wielenga. 2007. *Rich client programming: plugging into the netbeans platform*. Prentice Hall Press. 5.2

[31] J2EE Brain. 2013. Applications on eclipse. URL `http://www.j2eebrain.com/java-J2ee-applications-on-eclipse.html`. Accessed: May 2014. 7.23

[32] Statistic Brain. 2012. Computer sales statistics | statistic brain. URL `http://www.statisticbrain.com/computer-sales-statistics/`. Accessed: May 2014. 1.1

[33] Tonia De Bruin, Ronald Freeze, Uday Kaulkarni, and Michael Rosemann. 2005. Understanding the main phases of developing a maturity assessment model. In B Campbell, J Underwood, and D Bunker, editors, *Australasian Conference on Information Systems (ACIS)*, pages 8–19, Australia, New South Wales, Sydney. Australasian Chapter of the Association for Information Systems. URL `http://eprints.qut.edu.au/25152/`. 10

[34] Joop FLM Brukx and Ger L Wackers. 2005. Vulnerability profiling in complex socio-technological systems. *ECCON 2005*. 3.3

[35] Erik Brynjolfsson. 1993. The productivity paradox of information technology. *Communications of the ACM*, 36(12):66–77. 3.3

[36] Sabine Buckl, Markus Buschle, Pontus Johnson, Florian Matthes, and Christian M Schweda. 2011. A meta-language for enterprise architecture analysis. In *Enterprise, Business-Process and Information Systems Modeling*, pages 511–525. Springer. 5.3

[37] Sabine Buckl, Alexander M Ernst, Josef Lankes, Christian M Schweda, and André Wittenburg. 2007. Generating visualizations of enterprise architectures using model transformations. In *EMISA*, pages 33–46. Citeseer. 12.4

[38] Sabine Buckl, Florian Matthes, and Christian M Schweda. 2009. Classifying enterprise architecture analysis approaches. In *Enterprise Interoperability*, pages 66–79. Springer. 5.4

[39] Frank Budinsky. 2004. *Eclipse modeling framework: a developer's guide.* Addison-Wesley Professional. 7.3, 7.3

[40] Markus Buschle, Torsten Derlat, and Daniel Feller. 2009. Scenario-based architectural decision support within enterprise architectures. 6.2, 11.2

[41] Markus Buschle, Mathias Ekstedt, Sebastian Grunow, Matheus Hauder, Florian Matthes, and Sascha Roth. 2012. Automating enterprise architecture documentation using an enterprise service bus. In *AMCIS*. 12.2

[42] Markus Buschle, Hannes Holm, Teodor Sommestad, Mathias Ekstedt, and Khurram Shahzad. 2012. A tool for automatic enterprise architecture modeling. In *IS Olympics: Information Systems in a Diverse World*, pages 1–15. Springer. 6.2, 7.2, 9.6, 11.2, 12.2

[43] Markus Buschle, Pontus Johnson, and Khurram Shahzad. 2013. The enterprise architecture analysis tool–support for the predictive, probabilistic architecture modeling framework. *Association for Information Systems Conference, AMCIS 2013 Proceedings.* 3.2, 5.4, 6.2, 7.1, 11.2

[44] Markus Buschle, Johan Ullberg, Ulrik Franke, Robert Lagerström, and Teodor Sommestad. 2011. A tool for enterprise architecture analysis using the prm formalism. In *Information Systems Evolution*, pages 108–121. Springer. 5.2, 5.4, 6.2, 11.2

[45] Frank Buschmann, Kelvin Henney, and Douglas Schimdt. 2007. *Pattern-oriented Software Architecture: On Patterns and Pattern Language*, volume 5. John Wiley & Sons. 7.3

[46] Heiko Böck. 2011. *The Definitive Guide to NetBeans Platform 7.* Apress. 5.2

[47] Sven A Carlsson. 2007. Developing knowledge through is design science research. *Scandinavian Journal of Information Systems*, 19(2):75–86. 2.1

[48] Nicholas G Carr. 2003. It doesn't matter. *Educause Review*, 38:24–38. 1.1

[49] Chris K Carter and Robert Kohn. 1994. On gibbs sampling for state space models. *Biometrika*, 81(3):541–553. 5.4

[50] Per Cederqvist, Roland Pesch, *et al.* 1992. Version management with cvs. *Available online with the CVS package. Signum Support AB.* 12.1

[51] Moustafa Chenine, Johan Ullberg, Lars Nordström, Yiming Wu, and Göran Ericsson. 2013. A framework for wide area monitoring and control systems interoperability and cyber security analysis. submitted. 7.2

[52] Matthew Chu, Kyle Ingols, Richard Lippmann, Seth Webster, and Stephen Boyer. 2010. Visualizing attack graphs, reachability, and trust relationships with navigator. In *Proceedings of the Seventh International Symposium on Visualization for Cyber Security*, pages 22–33. ACM. 1.1

[53] David Cohen, Mikael Lindvall, and Patricia Costa. 2004. An introduction to agile methods. *Advances in computers*, 62:1–66. 6.1

[54] CIO Council. 1999. Federal enterprise architecture framework version 1.1. *Retrieved from*, 80. 3.1, 5.3

[55] Stephen Cranefield. 2006. Networked knowledge representation and exchange using uml and rdf. *Journal of Digital information*, 1(8). 9.8

[56] Nigel Cross. 2006. Design as a discipline. *Designerly Ways of Knowing*, pages 95–103. 2.1

[57] Krzysztof Czarnecki and Simon Helsen. 2003. Classification of model transformation approaches. In *Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture*, volume 45, pages 1–17. 7.3

[58] GC Dalton, Robert F Mills, John M Colombi, and Richard A Raines. 2006. Analyzing attack trees using generalized stochastic petri nets. In *Information Assurance Workshop, 2006 IEEE*, pages 116–123. IEEE. 5.6

[59] Arnaud de Borchgrave, Frank J Cilluffo, Sharon L Cardash, and Michèle M Ledgerwood. 2000. Cyber threats and information security. In *Meeting the*, volume 2. 1.1

[60] Deloitte. 2013. Deloitte announces 2013 technology fast 500 rankings. URL `http://www.deloitte.com/view/en_US/us/Industries/technology/5f0c69032e4fa310VgnVCM1000003156f70aRCRD.htm`. Accessed: May 2014. 1.1

[61] Jan LG Dietz. 2001. Demo: Towards a discipline of organisation engineering. *European Journal of Operational Research*, 128(2):351–363. 5.3

[62] Kyle Dunsire, Tim O'Neill, Mark Denford, and John Leaney. 2005. The abacus architectural approach to computer-based system and enterprise evolution. In *Engineering of Computer-Based Systems, 2005. ECBS'05. 12th IEEE International Conference and Workshops on the*, pages 62–69. IEEE. 3.2

[63] Charles M Eastman. 1999. *Building product models: computer environments, supporting design and construction*. CRC press. 1.1

[64] Mathias Ekstedt, Ulrik Franke, Pontus Johnson, Robert Lagerström, Teodor Sommestad, Johan Ullberg, and Markus Buschle. 2009. A tool for enterprise architecture analysis of maintainability. In *Software Maintenance and Reengineering, 2009. CSMR'09. 13th European Conference on*, pages 327–328. IEEE. 6.2, 11.2

[65] Gregor Engels, Jochen M Küster, Reiko Heckel, and Marc Lohmann. 2003. Model-based verification and validation of properties. *Electronic Notes in Theoretical Computer Science*, 82(7):133–150. 7.2

[66] Wilco Engelsman, Henk Jonkers, and Dick Quartel. 2011. Archimate® extension for modeling and managing motivation, principles, and requirements in togaf®. 3.1, 5.3

[67] PHISHING FACTS. 2006. Phishing mongers and posers. *Communications of the ACM*, 49(4):21. 3.3

[68] Gerald E. Farin, Josef Hoschek, and Myung-Soo Kim, editors. 2002. *Handbook of computer aided geometric design*. Elsevier, Amsterdam ; Boston, Mass. ISBN 0444511040. 4.3

[69] A Farooq, DH Owens, B Lokowandt, and B-M Pfeiffer. 1991. Model-based expert system for computer aided design of feedback controllers. In *Control 1991. Control'91., International Conference on*, pages 1246–1250. IET. 4.3, 4.4

[70] Matthias Farwick, Berthold Agreiter, Ruth Breu, Steffen Ryll, Karsten Voges, and Inge Hanschke. 2011. Automation processes for enterprise architecture management. In *Enterprise Distributed Object Computing Conference Workshops (EDOCW), 2011 15th IEEE International*, pages 340–349. IEEE. 4.1, 7.2, 12.2

[71] Matthias Farwick, Ruth Breu, Matheus Hauder, Sascha Roth, and Florian Matthes. 2013. Enterprise architecture documentation: Empirical analysis of information sources for automation. In *System Sciences (HICSS), 2013 46th Hawaii International Conference on*, pages 3868–3877. IEEE. 12.2

[72] Fortune. 2014. 7 fastest-growing tech companies - baidu (1) - FORTUNE. URL `http://money.cnn.com/gallery/technology/2013/08/29/fastest-growing-tech-companies-2013.fortune/index.html?iid=FGCos_sp_lead2`. Accessed: May 2014. 1.1

[73] The Apache Software Foundation. 2013. Math - commons math: The apache commons mathematics library. URL `http://commons.apache.org/proper/commons-math/`. Accessed: May 2014. 7.3

[74] The Apache Software Foundation. 2014. Maven. URL `http://maven.apache.org/`. Accessed: May 2014. 12.1

[75] The Eclipse Foundation. 2014. Eclipse modeling framework 2.9. documentation. URL `http://download.eclipse.org/modeling/emf/emf/javadoc/2.9.0/org/eclipse/emf/ecore/package-summary.html`. Accessed: May 2014. 7.28

[76] The Eclipse Foundation. 2014. Eclipse modeling framework project - EMF - home. URL `http://www.eclipse.org/modeling/emf/`. Accessed: May 2014. 5.2

[77] Ulrich Frank. 2011. The memo meta modelling language (mml) and language architecture. Technical report, ICB-Research Report. 5.3

[78] Ulrich Frank, David Heise, Heiko Kattenstroth, and Hanno Schauer. 2008. Designing and utilising business indicator systems within enterprise models-outline of a method. In *MobIS*, pages 89–105. 5.3

[79] Ulrik Franke, Mathias Ekstedt, Robert Lagerström, Jan Saat, and Robert Winter. 2010. Trends in enterprise architecture practice–a survey. In *Trends in Enterprise Architecture Research*, pages 16–29. Springer. 3.3

[80] Ulrik Franke, Pontus Johnson, and Johan König. 2013. An architecture framework for enterprise it service availability analysis. *Software & Systems Modeling*, pages 1–29. 9.7

[81] Ulrik Franke, Teodor Sommestad, Mathias Ekstedt, and Pontus Johnson. 2008. Defense graphs and enterprise architecture for information assurance analysis. Technical report, DTIC Document. 5.6, 5.6

[82] Rune Fredriksen, Monica Kristiansen, Bjørn Axel Gran, Ketil Stølen, Tom Arthur Opperud, and Theo Dimitrakos. 2002. The coras framework for a model-based risk management process. In *Computer Safety, Reliability and Security*, pages 94–105. Springer. 5.6

[83] Sanford Friedenthal, Alan Moore, and Rick Steiner. 2011. *A practical guide to SysML: the systems modeling language*. Elsevier. 7.2

[84] Shudi Gao, C Michael Sperberg-McQueen, Henry S Thompson, Noah Mendelsohn, David Beech, and Murray Maloney. 2009. W3c xml schema definition language (xsd) 1.1 part 1: Structures. *W3C Candidate Recommendation*, 30. 7.2

[85] Aditya Garg, Rick Kazman, and Hong-Mei Chen. 2006. Interface descriptions for enterprise architecture. *science of Computer Programming*, 61(1):4–15. 7.2

[86] Gartner. 2007. It spending and staffing survey, western europe, 2006-2007. *Gartner Research*. 3.3

[87] Gartner. 2011. Enterprise architecture tools are positioned to deliver business value. Accessed: May 2014. 1.1

[88] Gartner. 2011. Gartner assessment of enterprise architecture tool capabilities. URL `http://my.gartner.com/portal/server.pt?open=512&objID=260&mode=2&PageID=3460702&docCode=211294&ref=docDisplay`. Accessed: May 2014. 1.1, 3.2, 4, 10.1

[89] Gartner. 2011. Understanding the eight critical capabilities of enterprise architecture tools. URL `http://my.gartner.com/portal/server.pt?open=512&objID=260&mode=2&PageID=3460702&resId=1622120&ref=QuickSearch&sthkw=Understanding+the+Eight+Critical+Capabilities+of+Enterprise+Architecture+Tools`. Accessed: May 2014. 4.1, 4.1, 10.1

[90] Gartner. 2013. Hype cycle for enterprise architecture, 2013. URL `http://my.gartner.com/portal/server.pt?open=512&objID=260&mode=2&PageID=3460702&docCode=248891&ref=docDisplay`. Accessed: May 2014. 1.1

[91] Gartner. 2013. Research by topic. URL `http://www.gartner.com/it/products/research/topics/topics.jsp#ITM`. Accessed: May 2014. 1.1

[92] David Gilbert. 2002. The jfreechart class library. *Developer Guide. Object Refinery*. 7.3

[93] Stephen Gilmore, László Gönczy, Nora Koch, Philip Mayer, Mirco Tribastone, and Dániel Varró. 2011. Non-functional properties in the model-driven development of service-oriented systems. *Software & Systems Modeling*, 10 (3):287–311. 10.3

[94] Martin Gogolla and Mark Richters. 2002. Expressing uml class diagrams properties with ocl. In *Object Modeling with the OCL*, pages 85–114. Springer. 5.3

[95] Jaap Gordijn and Hans Akkermans. 2001. Designing and evaluating e-business models. *IEEE intelligent Systems*, 16(4):11–17. 3.3

[96] Shirley Gregor. 2006. The nature of theory in information systems. *MIS Quarterly*, 30(3):611–642. 2.1

[97] Shirley Gregor and David Jones. 2004. The formulation of design theories for information systems. In *Constructing the Infrastructure for the Knowledge Economy*, pages 83–93. Springer. 2.1

[98] DoDAF Architectures Framework Working Group *et al.* 2009. Dodaf architecture framework version 2.0. *Department of Defense United States*. 1.1

[99] Object Managment Group. 1999. Unified modeling language, UML 1.3. 5.3

[100] Object Managment Group. 2003. Object constraint language (OCL). OMG document ptc/03-10-14. *Specification (2003)*. 5.3

[101] Object Managment Group. 2008. Meta object facility (mof) 2.0 query/view/transformation specification. *Final Adopted Specification (November 2005)*. 5.3, 7.3

[102] Object Managment Group. 2014. Unified modeling language, UML 2.4.1. URL `http://www.omg.org/spec/UML/2.4.1/`. Accessed: May 2014. 5.3, 5.3, 7.3

[103] The Open Group. 2013. ArchiMate 2.1 specification. URL `http://pubs.opengroup.org/architecture/archimate2-doc/`. Accessed: May 2014. 5.3

[104] T.O. Group. 2013. *ArchiMate 2.1 Specification*. The Open group series. Haren Publishing, Van. ISBN 9789401805094. URL `http://books.google.lu/books?id=w3deAgAAQBAJ`. 3.1

[105] Olivier Gruber, BJ Hargrave, Jeff McAffer, Pascal Rapicault, and Thomas Watson. 2005. The eclipse 3.0 platform: adopting osgi technology. *IBM Systems Journal*, 44(2):289–299. 7.3

[106] Sebastian Grunow, Florian Matthes, and Sascha Roth. 2013. Towards automated enterprise architecture documentation: Data quality aspects of sap pi. In *Advances in Databases and Information Systems*, pages 103–113. Springer. 12.2

[107] Inge Hanschke. 2009. *Strategic IT Management: A Toolkit for Enterprise Architecture Management*. Springer. 1.1

[108] V. Haren and Van Haren Publishing. 2012. *ArchiMate 2. 0 Specification*. The Open Group. Van Haren Publishing. ISBN 9789087536923. URL `http://books.google.lu/books?id=THB_tgAACAAJ`. 3.1

[109] Van Haren. 2011. *TOGAF Version 9.1*, 10th edition. Van Haren Publishing. ISBN 9087536798, 9789087536794. 3.1

[110] Robert Harris and Rob Warner. 2004. *The definitive guide to SWT and JFace*. Apress. 7.3

[111] R. Harrison and The Open Group. 2007. *TOGAF Version 8.1.1 Enterprise Edition*. Togaf Series. Van Haren Publishing. ISBN 9789087530938. URL `http://books.google.se/books?id=a3MPmyUlLHUC`. 3.1

[112] Yujing He. 2006. Comparison of the modeling languages alloy and uml. In *Software Engineering Research and Practice*, pages 671–677. Citeseer. 5.3

[113] David R Heffelfinger. 2005. Getting started with jasperreports. 7.3

[114] Alan R Hevner, Salvatore T March, Jinsoo Park, and Sudha Ram. 2004. Design science in information systems research. *MIS quarterly*, 28(1):75–105. 2.1, 2.1, 2.1

[115] Rich Hilliard. 2000. Ieee-std-1471-2000 recommended practice for architectural description of software-intensive systems. *IEEE, http://standards. ieee. org*. 7.2

[116] Rich Hilliard *et al.* 2001. Viewpoint modeling. In *Proceedings of 1st ICSE Workshop on Describing Software Architecture with UML*. 7.2

[117] Hannes Holm. 2013. A large-scale study of the time required to compromise a computer system. *IEEE Transactions on Dependable and Secure Computing*, 11. 1.1

[118] Hannes Holm, Markus Buschle, Robert Lagerström, and Mathias Ekstedt. 2012. Automatic data collection for enterprise architecture models. *Software & Systems Modeling*, pages 1–17. 7.2, 9.6

[119] Hannes Holm, Khurram Shahzad, Markus Buschle, and Mathias Ekstedt. 2014. P2cysemol : Predictive, probabilistic cyber security modeling language. To be published. 5.5, 5.6, 5.6, 7.2, 7.2, 8.2, 8.1, 9.3, 9.8, 9.7, 11.2, 11.3

[120] Michael Howard and David LeBlanc. 2009. *Writing secure code*. O'Reilly Media, Inc. 5.6

[121] John Hunt. 2006. Feature-driven development. *Agile Software Construction*, pages 161–182. 6.1, 6.1

[122] Maria-Eugenia Iacob and Henk Jonkers. 2006. Quantitative analysis of enterprise architectures. In *Interoperability of Enterprise Software and Applications*, pages 239–252. Springer. 3.2

[123] Maria-Eugenia Iacob and Henk Jonkers. 2007. Quantitative analysis of service-oriented architectures. *International Journal of Enterprise Information Systems (IJEIS)*, 3(1):42–60. 5.3

[124] IBM. 2014. IBM - rational system architect. URL `http://www-03.ibm.com/software/products/en/ratisystarch`. Accessed: May 2014. 3.2

[125] Mega International. 2014. MEGA system blueprint | MEGA. URL `http://www.mega.com/en/product/mega-system-blueprint`. Accessed: May 2014. 5.3

[126] Alessio Ishizaka and Philippe Nemery. 2013. Multi-attribute utility theory. *Multi-Criteria Decision Analysis: Methods and Software*, pages 81–113. 3.3

[127] Stanislav Ivanov. 2011. A master thesis on porting the enterprise architecture analysis tool to eclipse modeling project. 5.2

[128] Daniel Jackson. 2002. Alloy: a lightweight object modelling notation. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 11(2): 256–290. 5.3

[129] Daniel Jackson. 2012. *Software Abstractions: logic, language, and analysis*. MIT press. 5.3

[130] Ivar Jacobson. 1992. *Object-oriented software engineering: a use case driven approach*. Pearson Education. 5.3

[131] Somesh Jha, Oleg Sheyner, and Jeannette Wing. 2002. Two formal analyses of attack graphs. In *Computer Security Foundations Workshop, 2002. Proceedings. 15th IEEE*, pages 49–63. IEEE. 5.6

[132] Erik Johansson. 2005. *Assessment of Enterprise Information Security - How to make it Credible and Efficient*. PhD thesis, Royal Institute of Technology (KTH). TRITA-ICS-0502. 6.2

[133] P. Johnson and M. Ekstedt. 2007. *Enterprise Architecture: Models and Analyses for Information Systems Decision Making*. Lightning Source Incorporated. ISBN 9789144027524. URL `http://books.google.lu/books?id=2LdxPQAACAAJ`. 3.2

[134] Pontus Johnson, Maria Eugenia Iacob, Margus Välja, Marten van Sinderen, Christer Magnusson, and Tobias Ladhe. 2013. Business model risk analysis: Predicting the probability of business network profitability. In *Enterprise Interoperability*, pages 118–130. Springer. 8.3, 8.1, 11.3

[135] Pontus Johnson, Erik Johansson, Teodor Sommestad, and Johan Ullberg. 2007. A tool for enterprise architecture analysis. In *Enterprise Distributed Object Computing Conference, 2007. EDOC 2007. 11th IEEE International*, pages 142–142. IEEE. 6.2, 11.2

[136] Pontus Johnson, Robert Lagerström, Mathias Ekstedt, and Magnus Österlind. 2012. It management with enterprise architecture. 3.2, 3.3, 3.3, 7.2, 8.3, 8.1, 8.4, 9.4, 9.7, 9.9, 11.3

[137] Pontus Johnson, Robert Lagerström, Per Närman, and Mårten Simonsson. 2007. Enterprise architecture analysis with extended influence diagrams. *Information Systems Frontiers*, 9(2-3):163–180. 3.2, 5.4

[138] Pontus Johnson, Robert Lagerström, Per Närman, and Mårten Simonsson. 2007. System quality analysis with extended influence diagrams. In *CSMR 2007 Workshop and Special Session papers*. 3.2, 4.2, 5.3

[139] Pontus Johnson, Lars Nordström, and Robert Lagerström. 2007. Formalizing analysis of enterprise architecture. In *Enterprise Interoperability*, pages 35–44. Springer. 3.1, 5.4

[140] Pontus Johnson, Johan Ullberg, Markus Buschle, Ulrik Franke, and Khurram Shahzad. 2013. P2amf: Predictive, probabilistic architecture modeling framework. In *Enterprise Interoperability*, pages 104–117. Springer Berlin Heidelberg. 5.4, 5.4, 5.7

[141] Jan Jürjens. 2002. Umlsec: Extending uml for secure systems development. In *UML 2002 The Unified Modeling Language*, pages 412–425. Springer. 5.6

[142] Ralph L Keeney and Howard Raiffa. 1993. *Decisions with multiple objectives: preferences and value trade-offs*. Cambridge university press. 3.3

[143] Daphne Koller. 1999. Probabilistic relational models. In *Inductive logic programming*, pages 3–13. Springer. 5.4, 6.2

[144] Johan König, Ulrik Franke, and Lars Nordstrom. 2010. Probabilistic availability analysis of control and automation systems for active distribution networks. In *Transmission and Distribution Conference and Exposition, 2010 IEEE PES*, pages 1–8. IEEE. 5.3

[145] Igor Kotenko and Mikhail Stepashkin. 2006. Attack graph based evaluation of network security. In *Communications and Multimedia Security*, pages 216–227. Springer. 5.6, 12.2

[146] Klaus Krogmann, Christian M Schweda, Sabine Buckl, Michael Kuperberg, Anne Martens, and Florian Matthes. 2009. Improved feedback for architectural performance prediction using software cartography visualizations. In *Architectures for Adaptive Software Systems*, pages 52–69. Springer. 12.4

[147] Manfred Kudlek. 2005. Probability in petri nets. *Fundamenta Informaticae*, 67(1):121–130. 5.3

[148] Stephan Kurpjuweit and Robert Winter. 2007. Viewpoint-based meta model engineering. In *EMISA*, volume 143, page 2007. 3.2

[149] Robert Lagerström, Ulrik Franke, Pontus Johnson, and Johan Ullberg. 2009. A method for creating enterprise architecture metamodels–applied to systems modifiability analysis. *International Journal of Computer Science and Applications*, 6(5):89–120. 5.3

[150] Robert Lagerström, Pontus Johnson, and Per Närman. 2007. Extended influence diagram generation. In *Enterprise Interoperability II*, pages 599–602. Springer. 5.4

[151] Ralph Langner. 2011. Stuxnet: Dissecting a cyberwarfare weapon. *Security & Privacy, IEEE*, 9(3):49–51. 3.3

[152] Joscf Lankes, Florian Matthes, and André Wittenburg. 2005. Softwarekartographie: systematische darstellung von anwendungslandschaften. In *Wirtschaftsinformatik 2005*, pages 1443–1462. Springer. 12.4

[153] M Lankhorst. 2009. Enterprise architecture at work: Modelling, communication and analysis. 1.1, 3.1, 3.2, 5.3, 5.5

[154] Kathryn Blackmond Laskey. 1995. Sensitivity analysis for probability assessments in bayesian networks. *Systems, Man and Cybernetics, IEEE Transactions on*, 25(6):901–909. 12.2

[155] Jong Seok Lee, Jan Pries-Heje, and Richard Baskerville. 2011. Theorizing in design science research. In *Service-Oriented Perspectives in Design Science Research*, pages 1–16. Springer. 2.1

[156] Edward Lewis. 2011. Annl: The tool for planning for viable enterprises. In *CAiSE Forum*, pages 121–130. 3.2

[157] Yu Liu and Hong Man. 2005. Network vulnerability assessment using bayesian networks. In *Defense and Security*, pages 61–71. International Society for Optics and Photonics. 5.6

[158] Torsten Lodderstedt, David Basin, and Jürgen Doser. 2002. Secureuml: A uml-based modeling language for model-driven security. In *UML 2002 The Unified Modeling Language*, pages 426–441. Springer. 5.6

[159] Kathleen N Lohr, Neil K Aaronson, Jordi Alonso, M Audrey Burnam, Donald L Patrick, Edward B Perrin, and James S Roberts. 1996. Evaluating quality-of-life and health status instruments: development of scientific review criteria. *Clinical therapeutics*, 18(5):979–992. 10.1

[160] Rafael Lotufo, Steven She, Thorsten Berger, Krzysztof Czarnecki, and Andrzej Wąsowski. 2010. Evolution of the linux kernel variability model. In *Software Product Lines: Going Beyond*, pages 136–150. Springer. 12.1

[161] Scott M Lynch. 2007. Modern model estimation part 1: Gibbs sampling. In *Introduction to Applied Bayesian Statistics and Estimation for Social Scientists*, pages 77–105. Springer. 5.4

[162] David A. Madsen. 2012. *Engineering drawing & design*, 5th ed edition. Delmar, Cengage Learning, Clifton Park, NY. ISBN 9781111309572. 4.3

[163] Miroslaw Malek, Bratislav Milic, and Nikola Milanovic. 2008. Analytical availability assessment of it services. In *Service Availability*, pages 207–224. Springer. 1.1

[164] Ignacio J Martinez-Moyano, SH Conrad, Eliot H Rich, and David F Andersen. 2006. Modeling the emergence of insider threat vulnerabilities. In *Simulation Conference, 2006. WSC 06. Proceedings of the Winter*, pages 562–568. IEEE. 3.3

[165] José Ramón San Cristóbal Mateo. 2012. Multi-attribute utility theory. In *Multi Criteria Analysis in the Renewable Energy Industry*, pages 63–72. Springer. 3.3

[166] Florian Matthes, Sabine Buckl, Jana Leitel, and Christian M Schweda. 2008. *Enterprise Architecture Management Tool Survey 2008*. Techn. Univ. München. 1.1, 3.2, 4, 4.1, 5.2, 7.2, 9.6, 10.1, 13

[167] Jeff McAffer, Jean-Michel Lemieux, and Chris Aniszczyk. 2010. *Eclipse rich client platform*. Addison-Wesley Professional. 5.2, 6.2, 7.3, 7.3

[168] Jeff McAffer, Paul VanderLei, and Simon Archer. 2010. *OSGi and Equinox: Creating highly modular Java systems*. Addison-Wesley Professional. 7.3

[169] Alan McLucas and Ed Lewis. 2008. A multi-methodology approach to addressing ict skill shortages in a government organization: Integration of system dynamics modeling and risk management. In *Proceedings of the 2008 International Conference of the System Dynamics Society*. 3.2

[170] Lucas O Meertens, Maria-Eugenia Iacob, Lambert JM Nieuwenhuis, MJ Van Sinderen, Henk Jonkers, and D Quartel. 2012. Mapping the business model canvas to archimate. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pages 1694–1701. ACM. 5.3

[171] KS Metaxiotis, Dimitris Askounis, and John Psarras. 2002. Expert systems in production planning and scheduling: a state-of-the-art survey. *Journal of Intelligent Manufacturing*, 13(4):253–260. 4.3, 4.5

[172] Microsoft. 2005. Lab 1: Implementing user management and authentication. URL `http://msdn.microsoft.com/en-us/library/ee810314(v=cs.20).aspx`. Accessed: May 2014. 12.1

[173] Jason Milletary and CERT Coordination Center. 2005. Technical trends in phishing attacks. *Retrieved December*, 1:2007. 3.3

[174] Henry Mintzberg. 1979. *The structuring of organization: A synthesis of the research*. Prentice-Hall. 3.3

[175] Audris Mockus. 2009. Amassing and indexing a large sample of version control systems: Towards the census of public source code history. In *Mining Software Repositories, 2009. MSR'09. 6th IEEE International Working Conference on*, pages 11–20. IEEE. 12.1

[176] OPNET Modeler. 2009. Opnet technologies inc. 1.1

[177] Robert Muetzelfeldt and Jon Massheder. 2003. The simile visual modelling environment. *European Journal of Agronomy*, 18(3):345–358. 12.2

[178] William Nagel. 2005. *Subversion Version Control: Using the Subversion Version Control System in Development Projects*. Prentice Hall PTR. 12.1

[179] K. Lalit Narayan, K. Mallikarjuna Rao, and M. M. M Sarcar. 2008. *Computer aided design and manufacturing*. Prentice-Hall of India, New Delhi. ISBN 9788120333420 812033342X. 4.3

[180] Per Närman, Markus Buschle, and Mathias Ekstedt. 2013. An enterprise architecture framework for multi-attribute information systems analysis. *Software & Systems Modeling*, pages 1–32. 8.3, 8.1, 10.3, 11.3

[181] Per Närman, Markus Buschle, Johan König, and Pontus Johnson. 2010. Hybrid probabilistic relational models for system quality analysis. In *Enterprise Distributed Object Computing Conference (EDOC), 2010 14th IEEE International*, pages 57–66. IEEE. 5.4, 7.2

[182] Per Närman, Ulrik Franke, Johan König, Markus Buschle, and Mathias Ekstedt. 2014. Enterprise architecture availability analysis using fault trees and stakeholder interviews. *Enterprise Information Systems*, 8(1):1–25. URL `http://dx.doi.org/10.1080/17517575.2011.647092`. 3.2, 5.3, 5.5, 8.3, 8.1, 11.3

[183] Per Närman, Hannes Holm, David Höök, Nicholas Honeth, and Pontus Johnson. 2012. Using enterprise architecture and technology adoption models to predict application usage. *Journal of Systems and Software*, 85(8):1953–1967. 8.3, 8.1, 11.3

[184] Per Närman, Hannes Holm, Pontus Johnson, Johan König, Moustafa Chenine, and Mathias Ekstedt. 2011. Data accuracy assessment using enterprise architecture. *Enterprise Information Systems*, 5(1):37–58. 8.3, 8.1, **??**, 9.6, 10.3, 11.3

[185] Per Närman, Pontus Johnson, Mathias Ekstedt, Moustafa Chenine, and Johan König. 2009. Enterprise architecture analysis for data accuracy assessments. In *Enterprise Distributed Object Computing Conference, 2009. EDOC'09. IEEE International*, pages 24–33. IEEE. 5.3

[186] Pia Närman, Pontus Johnson, and Liv Gingnell. 2014. Using enterprise architecture to analyze how organizational structure impact efficiency and quality of products and services. To be published. 8.3, 8.1, 11.3

[187] Nada Nassar, Nancy Helou, and Chantal Madi. 2013. Predicting falls using two instruments (the hendrich fall risk model and the morse fall scale) in an acute care setting in lebanon. *Journal of clinical nursing*. 10.1

[188] Steven Noel and Sushil Jajodia. 2004. Managing attack graph complexity through visual hierarchical aggregation. In *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, pages 109–118. ACM. 5.6

[189] Novell. 2014. Novell doc: OES 2 SP3: planning and implementation guide - linux user management: Access to linux for eDirectory users. URL `http://www.novell.com/documentation/oes2/oes_implement_lx/data/lum.html`. Accessed: May 2014. 12.1

[190] William L Oberkampf and Timothy G Trucano. 2002. Verification and validation in computational fluid dynamics. *Progress in Aerospace Sciences*, 38 (3):209–272. 7.2

[191] Department of Defense Architecture Framework Working Group *et al.* 2009. Department of defense architecture framework version 1.0. *Volume I: Definitions and Guidelines, February*, 9. 3.1

[192] Bureau of Economic Analysis. 2007. National economic accounts, table: 5.5.6. real private fixed investment in equipment and software by type, chained dollars. URL `http://www.bea.gov`. Accessed: May 2014. 3.3

[193] Pragmatic Enterprise Family of Frameworks. 2013. PEFF - enterprise frameworks. URL `http://www.pragmaticef.com/frameworks.htm`. Accessed: May 2014. 3.1

[194] Department of the Treasury Chief Information Officer Council. 2000. Treasury enterprise architecture framework version 1. 3.1

[195] Philipp Offermann, Sören Blom, Olga Levina, and Udo Bub. 2010. Proposal for components of method design theories. *Business & Information Systems Engineering*, 2(5):295–304. 2.1, 2.1

[196] Philipp Offermann, Olga Levina, Marten Schönherr, and Udo Bub. 2009. Outline of a design science research process. In *Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology*, page 7. ACM. 2.1

[197] Oracle. 2009. Oracle buys sun - oracle press release. URL `http://www.oracle.com/us/corporate/press/018363`. Accessed: May 2014. 5.2

[198] Oracle. 2010. Oracle supplier management implementation and administration guide. URL `http://docs.oracle.com/cd/E18727_01/doc.121/e16533/T553628T553633.htm`. Accessed: May 2014. 12.1

[199] Oracle. 2014. Learn about java technology. URL `http://www.java.com/en/about/`. Accessed: May 2014. 7.3

[200] Oracle. 2014. NetBeans. URL `https://netbeans.org/`. Accessed: May 2014. 5.2

[201] Oracle. 2014. NetBeans IDE - swing GUI builder (matisse) features. URL `https://netbeans.org/features/java/swing.html`. 5.2

[202] Oracle. 2014. NetBeans visual library. URL `https://platform.netbeans.org/graph/`. Accessed: May 2014. 5.2

[203] Oracle. 2014. Oracle JDeveloper - official home page. URL `http://www.oracle.com/technetwork/developer-tools/jdev/overview/index.html`. Accessed: May 2014. 5.2

[204] Magnus Osterlind, Pontus Johnson, Kiran Karnati, Robert Lagerström, and Margus Valja. 2013. Enterprise architecture evaluation using utility theory. In *Enterprise Distributed Object Computing Conference Workshops (EDOCW), 2013 17th IEEE International*, pages 347–351. IEEE. 7.3, 8.3, 8.1, 9.7, 11.3, 12.2

[205] Magnus Österlind, Robert Lagerström, and Peter Rosell. 2012. Assessing modifiability in application services using enterprise architecture models–a case study. *Trends in Enterprise Architecture Research and Practice-Driven Research on Enterprise Transformation*, pages 162–181. 8.3, 8.1, 11.3

[206] Xinming Ou, Sudhakar Govindavajhala, and Andrew W Appel. 2005. Mulval: A logic-based network security analyzer. In *14th USENIX Security Symposium*, pages 1–16. 1.1

[207] Oxford University Press. 1989. *The Oxford English dictionary*, 2nd ed edition. Clarendon Press ; Oxford University Press, Oxford : Oxford ; New York. ISBN 9780198611868. 1

[208] Joseph Pamula, Sushil Jajodia, Paul Ammann, and Vipin Swarup. 2006. A weakest-adversary security metric for network configuration security analysis. In *Proceedings of the 2nd ACM workshop on Quality of protection*, pages 31–38. ACM. 5.6

[209] Konstantinos Pantazis. 2014. Enterprise architecture at the financial sector with the eaat tool. Master thesis. 8.3

[210] Ken Peffers, Tuure Tuunanen, Charles E Gengler, Matti Rossi, Wendy Hui, Ville Virtanen, and Johanna Bragge. 2006. The design science research process: a model for producing and presenting information systems research. In *Proceedings of the first international conference on design science research in information systems and technology (DESRIST 2006)*, pages 83–106. 2.1, 2.1

[211] Ken Peffers, Tuure Tuunanen, Marcus A Rothenberger, and Samir Chatterjee. 2007. A design science research methodology for information systems research. *Journal of management information systems*, 24(3):45–77. 2.1, 2.1

[212] Jan Pries-Heje, Richard Baskerville, and John R Venable. 2008. Strategies for design science research evaluation. *ECIS 2008 Proceedings*. 2.1

[213] The TREsPASS Project. 2014. The TREsPASS project: Technology-supported risk estimation by predictive assessment of socio-technical security. URL `http://www.trespass-project.eu/`. Accessed: May 2014. 3.1

[214] Whitney Quesenbery. 2003. The five dimensions of usability. *Content and complexity: Information design in technical communication*, pages 81–102. 9.1

[215] Jane Radatz, Anne Geraci, and Freny Katki. 1990. Ieee standard glossary of software engineering terminology. *IEEE Std*, 610121990:121990. 3.3, 3.3

[216] Thomas C Redman and A Blanton. 1997. *Data quality for the information age*. Artech House, Inc. 3.3

[217] Trygve Reenskaug. 1979. Models-views-controllers. *Technical note, Xerox PARC*, 32:55. 6.2

[218] Allan Reid and Cisco Systems, Inc. 2008. *Networking for home and small businesses: CCNA discovery learning guide*. Cisco Networking Academy Program series. Cisco Press, Indianapolis, Ind. ISBN 9781587132094. 12.1

[219] VE van Reijswoud and Jan LG Dietz. 1999. Demo modelling handbook. *Delft University of Technology, Dept. of Information Systems*. 5.3

[220] Mark Richters and Martin Gogolla. 2002. Ocl: Syntax, semantics, and tools. In *Object Modeling with the OCL*, pages 42–68. Springer. 5.3

[221] Gerold Riempp and Stephan Gieffers-Ankel. 2007. Application portfolio management: a decision-oriented view of enterprise architecture. *Information Systems and E-Business Management*, 5(4):359–378. 3.3

[222] Nelson S Rosa, Paulo RF Cunha, and George RR Justo. 2002. Process nfl: A language for describing non-functional properties. In *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on*, pages 3676–3685. IEEE. 10.3

[223] Peter Rosell. 2012. Enterprise architecture modeling of core administrative systems at kth : A modifiability analysis. Master's thesis, KTH, Industrial Information and Control Systems. 8.3

[224] Jeanne W Ross, Peter Weill, and David C Robertson. 2006. *Enterprise architecture as strategy: Creating a foundation for business execution*. Harvard Business Press. 3.3

[225] Sascha Roth, Matheus Hauder, Matthias Farwick, Ruth Breu, and Florian Matthes. 2013. Enterprise architecture documentation: Current practices and future directions. In *Wirtschaftsinformatik*, page 58. 4.1

[226] Dan Rubel. 2006. The heart of eclipse. *Queue*, 4(8):36–44. 5.2

[227] Dan Rubel, Jaime Wren, and Eric Clayberg. 2011. *The eclipse graphical editing framework (gef)*. Addison-Wesley Professional. 7.3

[228] James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, William E. Lorensen, *et al.* 1991. *Object-oriented modeling and design*, volume 199. Prentice hall Englewood Cliffs (NJ). 5.3

[229] I Sabuncuoglu and DL Hommertzheim. 1989. Expert simulation systems: recent developments and applications in flexible manufacturing systems. *Computers & Industrial Engineering*, 16(4):575–585. 4.3, 4.3

[230] Stuart Edward Schechter. 2004. *Computer security strength & risk: A quantitative approach*. PhD thesis, Citeseer. 5.6

[231] Jaap Schekkerman. 2003. *How to Survive in the Jungle of Enterprise Architecture Framework: Creating or Choosing an Enterprise Architecture Framework*. Trafford. ISBN 141201607X. 3.1

[232] Akos Schmidt and Dániel Varró. 2003. Checkvml: A tool for model checking visual modeling languages. In *«UML» 2003-The Unified Modeling Language. Modeling Languages and Applications*, pages 92–95. Springer. 5

[233] Bruce Schneier. 1999. Attack trees. *Dr. Dobbs journal*, 24(12):21–29. 5.6

[234] Marten Schöenherr. 2009. Towards a common terminology in the discipline of enterprise architecture. In *Service-Oriented Computing–ICSOC 2008 Workshops*, pages 400–413. Springer. 7.2

[235] David A Schum. 1994. *The evidential foundations of probabilistic reasoning*. Northwestern University Press. 9.2

[236] D Scott. 2009. How to assess your it service availability levels. *Gartner, Inc., Tech. Rep.* 3.3

[237] IBM Global Services. 1998. Improving systems availability. *IBM*. 3.3

[238] Glenn Shafer. 1976. *A mathematical theory of evidence*, volume 1. Princeton university press Princeton. 5.4

[239] Hanifa Shah and Mohamed El Kourdi. 2007. Frameworks for enterprise architecture. *It Professional*, 9(5):36–41. 1.1

[240] Jianlin Shi and Martin Törngren. 2005. An overview of uml2 and brief assessment from the viewpoint of embedded control systems development. *Rap. tech., Mechatronics Lab, Dpt. of Machine Design, Royal Institute of Technology, Stockholm*, 21. 5.3

[241] Anna Sidorova, Nicholas Evangelopoulos, Russell Torres, and Vess Johnson. 2013. It and organizations. In *A Survey of Core Research in Information Systems*, pages 33–49. Springer. 2.1

[242] Vladimir Silva. 2009. *Practical Eclipse Rich Client Platform Projects*. Apress. 7.3

[243] Kevin Lee Smith and Tom Graves. 2011. *An Introduction to PEAF: Pragmatic Enterprise Architecture Framework*. Pragmatic EA Limited. 5.3

[244] Manfred Soeffky. 1998. Data warehouse: Prozess-und systemmanagement. *IT Research, Höhenkirchen*. 5.1

[245] Powersim Software. 2014. Powersim - studio 9. URL `http://www.powersim.com/info/about/news/new-product-edition-studio-9-express/`. Accessed: May 2014. 3.2

[246] Teodor Sommestad, Mathias Ekstedt, and Hannes Holm. 2013. The cyber security modeling language: A tool for assessing the vulnerability of enterprise system architectures. *Systems Journal, IEEE*, 7(3):363–373. 5.3, 5.6, 8.2, 8.1, 11.3

[247] Teodor Sommestad, Mathias Ekstedt, and Pontus Johnson. 2010. A probabilistic relational model for security risk analysis. *Computers & Security*, 29 (6):659–679. 5.6, 5.6, 5.6

[248] Teodor Sommestad, Mathias Ekstedt, and Pontus Johnson. 2010. A probabilistic relational model for security risk analysis. *Computers & Security*, 29 (6):659–679. 8.2

[249] Jonathan Sprinkle, Bernhard Rumpe, Hans Vangheluwe, and Gabor Karsai. 2011. Metamodelling. In *Model-Based Engineering of Embedded Real-Time Systems*, pages 57–76. Springer. 7.3

[250] Dave Steinberg, Frank Budinsky, Ed Merks, and Marcelo Paternostro. 2008. *EMF: eclipse modeling framework*. Pearson Education. 5.2, 5.3, 6.2, 7.3

[251] Cook Steve and Daniels John. 1994. Designing object systems: object-oriented modelling with syntropy. 5.3

[252] Mark CJ Stoddart. 2004. Generalizability and qualitative research in a post-modern world. *Graduate Journal of Social Science*, 1(2):303–317. 10.3

[253] Christoph Stoermer, Felix Bachmann, and Chris Verhoef. 2003. Sacam: The software architecture comparison analysis method. Technical report, DTIC Document. 10.2

[254] Diane M Strong, Yang W Lee, and Richard Y Wang. 1997. Data quality in context. *Communications of the ACM*, 40(5):103–110. 3.3

[255] Sparx Systems. 2014. Enterprise architect. URL `http://www.sparxsystems.com/products/ea/`. Accessed: May 2014. 5.3

[256] Troux Technologies. 2014. Troux architect. URL `http://www.troux.com/products/troux_software/`. Accessed: May 2014. 3.2

[257] Tetsuo Tomiyama, Takashia Kiriyama, Hideaki Takeda, Deye Xue, and Hiroyuki Yoshikawa. 1989. Metamodel: a key to intelligent cad systems. *Research in engineering design*, 1(1):19–34. 4.4

[258] Ambrosio Toval, Víctor Requena, and José Luis Fernández. 2003. Emerging ocl tools. *Software and Systems Modeling*, 2(4):248–261. 5.3

[259] Johan Ullberg, Ulrik Franke, Markus Buschle, and Pontus Johnson. 2010. A tool for interoperability analysis of enterprise architecture models using pi-ocl. In *Enterprise Interoperability IV*, pages 81–90. Springer. 5.4, 6.2, 11.2

[260] Johan Ullberg, Pontus Johnson, and Markus Buschle. 2011. A modeling language for interoperability assessments. In *Enterprise Interoperability*, pages 61–74. Springer. 5.3, 5.4, 7.2, 8.3, 8.1, 8.4, 9.7, 11.3

[261] Vijay K Vaishnavi and William Kuechler Jr. 2007. *Design science research methods and patterns: innovating information and communication technology*. CRC Press. 2.1

[262] Carlos Valcarcel. 2004. *Eclipse kick start*. Sams. 7.3

[263] Margus Valja, Magnus Osterlind, Maria-Eugenia Iacob, Marten van Sinderen, and Pontus Johnson. 2013. Modeling and prediction of monetary and non-monetary business values. In *Enterprise Distributed Object Computing Conference (EDOC), 2013 17th IEEE International*, pages 153–158. IEEE. 8.3, 8.1, 11.3

[264] John Venable, Jan Pries-Heje, and Richard Baskerville. 2012. A comprehensive framework for evaluation in design science research. In *Design Science Research in Information Systems. Advances in Theory and Practice*, pages 423–438. Springer. 2.1

[265] TOGAF Version. 2009. 9, the open group architecture framework (togaf). *The Open Group*. 1.1

[266] Lingyu Wang, Sushil Jajodia, Anoop Singhal, and Steven Noel. 2010. k-zero day safety: Measuring the security risk of networks against unknown attacks. In *Computer Security–ESORICS 2010*, pages 573–587. Springer. 1.1

[267] Jos B Warmer and Anneke G Kleppe. 2003. *The object constraint language: getting your models ready for MDA*. Addison-Wesley Professional. 5.3

[268] Michel Wermelinger and Yijun Yu. 2008. Analyzing the evolution of eclipse plugins. In *Proceedings of the 2008 international working conference on Mining software repositories*, pages 133–136. ACM. 5.2

[269] Brian Whitworth and Michael Zaic. 2003. The wosp model: Balanced information system design and evaluation. *Communications of the Association for Information Systems*, 12. 10.3

[270] Leevar Williams, Richard Lippmann, and Kyle Ingols. 2008. *GARNET: A graphical attack graph and reachability network evaluation tool*. Springer. 1.1

[271] Robert Winter. 2008. Design science research in europe. *European Journal of Information Systems*, 17(5):470–475. 2.1

[272] Robert Winter and Joachim Schelp. 2008. Enterprise architecture governance: the need for a business-to-it approach. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 548–552. ACM. 3.3

[273] AJ Wright, D Bloomfield, and TJ Wiltshire. 1992. Building simulation and building representation: Overview of current developments. *Building Services Engineering Research and Technology*, 13(1):1–11. 4.3, 4.2

[274] Jian-Bo Yang. 2001. Rule and utility based evidential reasoning approach for multiattribute decision analysis under uncertainties. *European Journal of Operational Research*, 131(1):31–61. 5.4

[275] Robert K Yin. 2009. *Case study research: Design and methods*, volume 5. sage. 10.2, 10.3

[276] John A Zachman. 1987. A framework for information systems architecture. *IBM systems journal*, 26(3):276–292. 1.1, 3.1

[277] Marin Zec. 2014. Enterprise architecture visualization tool survey 2014. 3.2

[278] Ibrahim Zeid. 1991. *CAD/CAM theory and practice*. McGraw-Hill Higher Education. 4.3

List of publications

[1] M. Ekstedt, U. Franke, P. Johnson, R. Lagerström, T. Sommestad, J. Ullberg, and M. Buschle, "A tool for enterprise architecture analysis of maintainability," in *Software Maintenance and Reengineering, 2009. CSMR'09. 13th European Conference on.* IEEE, 2009, pp. 327–328.

[2] P. Närman, M. Buschle, J. König, and P. Johnson, "Hybrid probabilistic relational models for system quality analysis," in *Enterprise Distributed Object Computing Conference (EDOC), 2010 14th IEEE International.* IEEE, 2010, pp. 57–66.

[3] J. Ullberg, U. Franke, M. Buschle, and P. Johnson, "A tool for interoperability analysis of enterprise architecture models using pi-ocl," *Enterprise Interoperability IV*, pp. 81–90, 2010.

[4] U. Franke, O. Holschke, M. Buschle, P. Närman, and J. Rake-Revelant, "IT consolidation: An optimization approach," in *Enterprise Distributed Object Computing Conference Workshops (EDOCW), 2010 14th IEEE International.* IEEE, 2010, pp. 21–26.

[5] M. Buschle, J. Ullberg, U. Franke, R. Lagerström, and T. Sommestad, "A tool for enterprise architecture analysis using the prm formalism," *Information Systems Evolution*, pp. 108–121, 2011.

[6] R. Lagerström, T. Sommestad, M. Buschle, and M. Ekstedt, "Enterprise architecture management's impact on information technology success," in *System Sciences (HICSS), 2011 44th Hawaii International Conference on.* IEEE, 2011, pp. 1–10.

[7] M. Buschle and D. Quartel, "Extending the method of bedell for enterprise architecture valuation," in *Enterprise Distributed Object Computing Conference Workshops (EDOCW), 2011 15th IEEE International.* IEEE, 2011, pp. 370–379.

[8] S. Buckl, M. Buschle, P. Johnson, F. Matthes, and C. M. Schweda, "A metalanguage for enterprise architecture analysis," in *Enterprise, Business-Process*

and Information Systems Modeling. Springer Berlin Heidelberg, 2011, pp. 511–525.

[9] J. Ullberg, P. Johnson, and M. Buschle, "A modeling language for interoperability assessments," *Enterprise Interoperability*, pp. 61–74, 2011.

[10] M. Buschle, H. Holm, T. Sommestad, M. Ekstedt, and K. Shahzad, "A tool for automatic enterprise architecture modeling," in *IS Olympics: Information Systems in a Diverse World*. Springer, 2012, pp. 1–15.

[11] P. Närman, U. Franke, J. König, M. Buschle, and M. Ekstedt, "Enterprise architecture availability analysis using fault trees and stakeholder interviews," *Enterprise Information Systems*, no. ahead-of-print, pp. 1–25, 2012.

[12] M. Buschle, M. Ekstedt, S. Grunow, M. Hauder, F. Matthes, and S. Roth, "Automating enterprise architecture documentation using an enterprise service bus." in *AMCIS*, 2012.

[13] H. Holm, M. Buschle, R. Lagerström, and M. Ekstedt, "Automatic data collection for enterprise architecture models," *Software & Systems Modeling*, pp. 1–17, 2012.

[14] J. Ullberg, P. Johnson, and M. Buschle, "A language for interoperability modeling and prediction," *Computers in Industry*, vol. 63, no. 8, pp. 766–774, 2012.

[15] N. Honeth, M. Buschle, R. Lagerström, K. K. Sasi, and S. Nithin, "An extended archimate metamodel for microgrid control system architectures," in *:*, 2012.

[16] M. Buschle, M. Ekstedt, S. Grunow, M. Hauder, F. Matthes, and S. Roth, "Amcis 2012 proceedings," in *Proceedings of the 18th Americas Conference on Information Systems (AMCIS)*, 2012.

[17] P. Närman, M. Buschle, and M. Ekstedt, "An enterprise architecture framework for multi-attribute information systems analysis," *Software & Systems Modeling*, pp. 1–32, 2013.

[18] P. Johnson, J. Ullberg, M. Buschle, U. Franke, and K. Shahzad, "P2amf: Predictive, probabilistic architecture modeling framework," in *Enterprise Interoperability*. Springer Berlin Heidelberg, 2013, pp. 104–117.

[19] M. Buschle, P. Johnson, and K. Shahzad, "The enterprise architecture analysis tool–support for the predictive, probabilistic architecture modeling framework," in *Proceedings of the 19th Americas Conference on Information Systems (AMCIS)*, 2013.

[20] U. Franke, M. Buschle, and M. Österlind, "An experiment in sla decision-making," in *Economics of Grids, Clouds, Systems, and Services*. Springer International Publishing, 2013, pp. 256–267.

[21] H. Holm, K. Shahzad, M. Buschle, and M. Ekstedt, "P2cysemol : Predictive, probabilistic cyber security modeling language," 2014, to be published.

[22] P. Johnson, J. Ullberg, M. Buschle, U. Franke, and K. Shahzad, "An architecture modeling framework for probabilistic prediction," *Information Systems and e-Business Management*, pp. 1–28, 2014.