TU WIEN Informatics

A Generic Conceptual Model to Graph Transformation Framework and its Application for Enterprise Architecture Analysis

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieurin

in

066 926 Business Informatics

by

BSc Muhamed Smajevic
Registration Number 11742556

to the Faculty of Informatics

at the TU Wien

Advisor: Assistant Prof. Dipl.-Wirtsch.Inf.Univ. Dr.rer.pol. Dominik Bork

Vienna, 21st November, 2022

Muhamed Smajevic	Dominik Bork

Erklärung zur Verfassung der Arbeit

BSc Muhamed Smajevic

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 21. November 2022

Muhamed Smajevic

Acknowledgements

First of all, I want to thank God for providing me the possibility to be where I am now and to finish working on this thesis. I have spent more than two years writing and implementing necessary things, which is much more than it should be, and I can say I do not regret it at all. It was not an easy path, and I learned a lot. I met many interesting people and tasted how it is to work in academia and be a part of the research community. I worked mainly on the thesis in Austria but also did important parts in my home country Bosnia. This would not be possible to accomplish at all if there were no people around me to whom I wanted to express my gratitude. I will mention some of them here, but also, there are others I did not forget but are not mentioned here (they will know who they are). I want to tell you how incredibly grateful I am to have you.

I want to express my first and most profound appreciation to my mentor Dominik Bork. If I put aside all his expertise and essential feedback, I want to thank him for changing how I think and approach things. He was great motivation; he always tried to push me to gain more and made me feel important by bringing the best out of me, which I am most thankful for.

Secondly, I want to thank my family for their tremendous support. I want to thank them for supporting my decision to move to Vienna and do my master's there. I want to thank my parents, Meho and Hamida, for constantly asking me about my progress and creating a slight pressure to keep working and don't quit. I want to thank my brother Arslan and sister Emina for so many talks showing their trust in me.

Next, I want to express my gratitude to my best friend, Kenan. He constantly increased my confidence and was my destination to talk to whenever I had stressful situations.

I also want to thank Ahsena for her unique role in this journey.

Lastly, I want to thank my two bosses, Manfred and Florian, for offering new business opportunities in exchange for finishing my studies.

Now it is time for something new.

Abstract

Core assets of enterprise architecture (EA) are its models. They encapsulate enterprise structure and IT knowledge by providing a holistic view of an organization that can drive decision-making. For making good decisions, it is necessary to have an excellent overview and to know how to analyze the model to gain information that different business stakeholders need. From the perspective of history, EA has evolved a lot, and the models have become larger and more complex. Models' change in size and complexity requires other techniques to analyze the model since the analysis of models in their raw form can exceed human capabilities by far. Therefore current approaches focus on studying partial views over the model, thus providing a partial valuation. This thesis proposes a slightly different approach to cope with EA analysis. The method is based on transforming conceptual models into graphs and analyzing the model in a graph-based manner by using knowledge and possibilities from graph theory. To put this into the real world, we have developed a generic framework and implemented it in the prototype platform, which we call CM2KG. The platform allows analysts to upload different models created in industry-standard tools and transform them into a standardized graph format. From there on, it becomes possible for them to visualize and interact with the model using multiple techniques, query the whole model, and define custom functions that will provide valuable information. We discuss some basic examples for each part to scratch the surface of what is possible. We also put one existing application of EA analysis into the perspective of this approach by implementing queries that are used to identify smells in EA models.

Contents

CHAPTER 1

Introduction

In their work, Davoudi and Aliee state that *"Enterprises are complex, highly integrated systems comprised of processes, organizations, information, and supporting technologies, with multifaceted inter-dependencies and interrelationships across their boundaries"* [DA09]. Enterprise Architecture (EA) based on Holm et al. "describes the fundamental artifacts of business and IT as well as their interrelationships, typically through dimensions such as business, application, technology, and information" [Hol+12].

EA based on Österlind, Lagerström, and Rosell "has become an established discipline for business and IT management" [ÖLR12] with many initiatives. Dreyfus and Iyer claim that since "enterprises face a turbulent and competitive environment, they must continuously innovate and engage in multiple IT implementation projects" [DI06]. Buschle, Johnson, and Shahzad state that "the management of organizations and their IT systems require frequent decision-making" [BJS13]. Veneberg et al. claim that when "having to decide about strategy and investments, managers deal with complex situations that could influence the future of their organizations" [Ven+14].

In order to provide guidance for project decisions, Dreyfus and Iyer state that *"companies develop a designed architecture where subsequent projects' impact on existing applications, data, and technology can, and often does, create a gap between the emergent architecture and the designed architecture"* [DI06]. Buschle, Johnson, and Shahzad in their work claim that "these decisions can, for example, be whether an existing system, in order to provide the performance needed for future services, should be replaced, upgraded, or kept as it is" [BJS13] or "whether two applications should be integrated, used in the same fashion they have been used until now or whether the provided services should be outsourced" [BJS13]. For these decisions, Barbosa et al. state that "EA analysis is considered as a fundamental practice of Enterprise Architecture Modeling" [Bar+19].

When it comes to different approaches, Lantow et al. claim that "many analysis approaches have been proposed by researchers and current Enterprise Architecture Modeling (EA

Modeling) tools implement analysis functionalities where in practice, EA is often analyzed by using visualizations and are typically created using EA Modeling tools" [Lan+16]. Analyzing previous works, Srinivas, Gill, and Roach conclude that "traditional EA modeling and deliverables are composed of static diagrams or viewpoints that are presented to stakeholders and do not change with the changing nature of data" [SGR20]. Regarding the new analysis approaches, Lantow et al. then state that "modern analysis approaches should combine interactive visualizations with automated analysis techniques" [Lan+16].

If we summarize all the information from the different papers above, we can see how vital analysis is for businesses. However, how much is done in this area? While little research about mechanisms to classify, compare, or organize the existing EA analysis research can be found, Barbosa et al., in their paper [Bar+19], introduce a taxonomy for EA analysis research. The taxonomy introduces four main dimensions: EA scope, analysis concern, analysis technique, and modeling language.

Regarding the modeling language dimension, paper states that 80% of all papers lead to 4 modeling language groups: ArchiMate (ArchiMate)-based, combined models, graphs, own. Regarding the graphs group, Barbosa et al. describe that "in the graphs group, the EA are modeled as graphs, with their components and relations being represented by nodes and edges, respectively" [Bar+19]. Another dimension in the taxonomy is the analysis technique, which covers techniques and methods to perform EA analysis, taking into consideration another dimension, the analysis concern dimension.

Results show that "about 70% of the studies corresponded to the following five values for analysis technique: (Semi) Formalism Based, Metric-based, Probabilistic-based, Structural Analysis, and Visual Analysis" [Bar+19]. Moreover, since some techniques are highly dependant on which modeling language is used, researchers remodeled or transformed the models from ArchiMate modeling language to graph modeling language in order to get new insights.

1.1 Problem Definition

This section is dedicated to specifying the problem for this thesis. The main research questions are formulated. The research objective clarifies what this research wants to achieve. Also, the scope of the thesis is defined.

1.1.1 Problem Statement

In order to to provide the stakeholders with additional insights some researchers have incorporated graph-based approach in their analysis. For example, Garg, Kazman, and Chen, in their paper [GKC06], describe how a 3-tier architecture component diagram can be designed as a graph to provide stakeholders with the following results: a *system of systems* view, architectural analyses (reliability, redundancy), risk analysis. The same paper [GKC06] states that the system has already provided essential insights into enterprise architecture that was hitherto unavailable.

A specific tool (EA Builder) was developed by Aier in [Aie06] in order to create clusters based on graph structure where the clusters identified are candidates for service domains while the activities are candidates for services that have to encapsulate the existing IT systems. In order to aid decision-making, Österlind, Lagerström, and Rosell, in their work [ÖLR12], suggest changes in *Modifiability Analysis Tool* and present a case study in which these changes have been tested.

Authors Holschke et al. propose another idea in their paper [Hol+09] on using Enterprise Architecture models coupled with Bayesian Belief Networks to facilitate Failure Impact Analysis. Buschle, Johnson, and Shahzad in [BJS13] present the class model that allows investigating availability at enterprises and is built upon ArchiMate, which makes use of the fault tree formalism. Another approach by Wood et al. in their research [Woo+13] collects the different stakeholders in EA and puts them in the Social Network analysis context.

Similar to previous authors, Dreyfus and Iyer propose "Architecture as a social system" in [DI06] by emphasizing network analysis. Literature shows that many papers talk about EA analysis techniques involving graph theory. A more detailed discussion about the related work and the motivation for this thesis can be found in Sec. 3.1.

Coming back to different dimensions from EA analysis taxonomy ([Bar+19]) there are two cases. We have examples modeled in a graph modeling language supporting graph analysis as the first case. The second case represents all other modeling languages where only a specific subset of EA elements is transformed to a graph and analyzed as before. The problem is that there is no general approach that will provide different analysis techniques that graph modeling languages provide to other categories in modeling language dimension (ArchiMate, Combined, Own). In their big survey of EA analysis and network thinking, Santana, Fischbach, and Moura perform a broad literature review of EA network-based analysis [SFM16]. In their work they identify applied measures and main achievements and propose potential topics for future research. One of the missing parts and recommendations for future research is the tool development for EA analysis since most papers propose the measures only theoretically. Combining the problem of lacking application of graph-based analysis techniques to different modeling languages with the recommendation of building up the tool that will enable practical EA analysis with different measures, leads us to a motivation to explore this area and ask questions that are described in the next section.

1.1.2 Research Questions and Objectives

In this section, we formulate the research questions. Additionally, we define the respective research objectives for each question that will guide us to answer a question.

RQ1 Is it possible to accomplish a generic approach of conceptual model analysis based on model transformation and graph structures at all?

Objective 1.1 *Quantitative and qualitative analysis of the conceptual models overview.*
The objective is to do a review of related work with respect to the classification of existing metrics and algorithms for graph/network analysis.

Objective 1.2 *Concept definition to transform conceptual models into graph structures.*
The objective is to explore a new generic concept based on model transformation on meta^2-level to a graph structure.

RQ2 What is an appropriate means to automate graph-based analysis of Enterprise Architecture models?

Objective 2.1 *Automation of the graph-based analysis of conceptual models.*
The objective is to explore the ways to automate the analysis based on different technologies/libraries/frameworks for ADOxx (ADOxx)-based and Ecore (Ecore)-based modeling languages.

RQ3 Is it possible to detect smells in EA and UML models using graph-based analysis approach?

Objective 3.1 *Specific smell detection application in graph-based analysis.*
The objective is to research how the graph-based analysis approach can be used to detect smells in EA and UML models.

RQ4 How does the graph-based analysis of Enterprise Architecture models compare to model analysis techniques provided by State-of-the-Art Enterprise Architecture Management tools?

Objective 4.1 *Exploration of analysis options provided by State-of-the-art EAM tools overview.*
The objective is to explore the different options that some tools provide regarding the analysis. We will take Archi (Archi) that supports ArchiMate modeling, TEAM library developed in ADOxx, and yED (yED), which is already based on graph structure. We do this to include different sides into an overview. On the other hand, we also want to explore the industry standards and compare them with open source options.

1.1.3 Research Scope

The work aims to provide different analysis techniques currently supported for graph-based modeling languages to other categories in the modeling language dimension mentioned in EA analysis taxonomy. This will be done by developing a general concept to analyze the conceptual models based on many already developed and proven concepts in graph theory and social network analysis.

The idea behind this is to have transformation [Lan+12] of high-level meta-modeling to a graph structure and analyze a graph. This concept will be applied to two meta-models: Ecore [Ecl21a] and ADOxx [ADO21a]. Next, the models created using tools Archi and TOGAF based Enterprise Architecture Management (TEAM) [Bor+18], which provide instances of Ecore (developed by ArchiMate modeling language) and ADOxx respectively, will be used for analysis. These models will be transformed and analyzed, and the analysis will be compared with the analysis options provided by state-of-the-art tools. We want to find out what graph-based analysis enables or more efficiently solves than non-graph-based approaches.

Concerning this context, three examples are provided. Different concerns for Enterprise Architect covered in [Bar+19] like Cost Analysis, EA Change, and EA Alignment involve impact analysis, so one example is how the tools respond to this. Another example are the options to prioritize the importance of an entity in the Enterprise Architecture. The third example can be seen as identification and visualisation of sub-communities within the Enterprise Architecture.

To achieve this, the analysis of the transformed models will be based on two approaches: a quantitative and qualitative analysis. For quantitative analysis [DES17], for the sake of simplicity, standard graph measures and algorithms will be taken. The measures that will be considered are degree centrality, closeness centrality, Eigenvector centrality, PageRank centrality, connected components, local clustering [EK13] [BJT16].

On the other side, qualitative analysis will include: graph querying, graph visualization, different layouts based on size and color of nodes and edges concerning a different node or edge measures, and multi-layer visualization [CSQ08]. Furthermore, qualitative research to obtain information about human perception of the given comparison will be performed.

Lastly, some of the known issues in EA analysis would be considered how they can be solved using the graph-based analysis approach. This includes the detection of smells in EA models, which can be identified by profoundly examining the structure of the model and trying to find some pattern that would yield what is considered as smell. The scope is to show this also on a similar setup but the different domain. Such domain is the UML class diagram, where there much of discussion about detecting smells in the class diagram model.

1.2 Research Methodology

To conduct this research, we will use Design Science Research [HC10]. We start by setting a goal to produce an artifact in several iterations. After each iteration, this artifact should be evaluated. Regrading the methodology framework, we will use Design Science Research Methodology (DSRM) described in [HC10]. The Design Science (DS) *process includes six steps: problem identification and motivation; definition of the objectives for a solution, design, and development; demonstration; evaluation; and communication*[HC10]. Below are the descriptions on how the all the activities will be performed [BHM20].

Activity 1. Problem identification and motivation. This was already discussed in 1.1.

Activity 2. Define the objectives for a solution. This was already discussed in 1.1.2.

Activity 3. Design and development. Here we first design and develop *the transformation of the conceptual model to a graph.* The transformation concept from the conceptual model to a graph has to be developed along with two examples Ecore to GraphML (GraphML) and ADOxx to GraphML.

Next, we proceed with *graph analysis.* The method of graph analysis concept consists of the following steps:

(a) Quantitative analysis - graph measures have to be applied to a graph like: degree centrality, closeness centrality, Eigenvector centrality, PageRank centrality, connected components, local clustering.

(b) Qualitative analysis - different visualizations have to applied using library Gephi (Gephi)/Neo4j (Neo4j)/yED: graph querying, graph visualization, different layouts based on size and color of nodes and edges with respect to a different node or edge measure and multi-layer visualization.

And lastly, *smell detection.* Development of queries to detect EA models smells and UML class diagram model smells.

Activity 4. Demonstration. The artifacts will be demonstrated with multiple different ArchiMate, TEAM, and UML class diagram models.

Activity 5. Evaluation. Evaluation will show how different models are transformed and how graph-based approaches cope with EA model analysis. For research questions, we define the requirements that will be used for evaluation.

Activity 6. Communication. Since no actual practicing professionals are available, the communication will be provided through different research papers to academic audience and feedback will be gathered from there.

1.3 Significance of the Thesis

This thesis aims to create practical and theoretical contributions in enterprise architecture analysis and graph-based domains. The thesis will provide some general knowledge about the graph-based analysis approach. It will try to answer if the general analysis approach is feasible. It will try to create artifacts that different stakeholders can use to perform different actions. Introduction to the State-of-the-Art tools will also be provided to a reader. Moreover, the project will create some insights and hopefully open possible areas for further research.

1.4 Thesis Outline

The rest of the thesis is divided into six chapters. It starts with the background knowledge about the terms and topics used throughout the thesis. The next chapter is about the related work, which will describe different studies and some state-of-the-art tools. This will summarize the gaps and lead to a better understanding of the motivation for this thesis. The next chapter is devoted to designing the general framework used for analysis. The fifth chapter is about implementing the prototype platform using the framework developed in chapter four. After the platform is developed, we provide an evaluation of research objectives in chapter six. The last chapter will conclude and answer the research questions defined in the first chapter.

1.5 Publications Based on This Thesis

In parallel to writing this thesis, the author of the thesis in the co-author role was involved in several publications based on topics from this thesis. The first paper [SB21b] was a joint work of Smajevic and Bork named *"Towards Graph-based Analysis of Enterprise Architecture Models"* which was published in the 40th International Conference on Conceptual Modeling. The second paper [SHB21] named *"Using Knowledge Graphs to Detect Enterprise Architecture Smells"* was a joint work of Smajevic, Hacks, and Bork and was presented in 14th IFIP Working Conference, PoEM 2021, Riga, Latvia, November 24-26, 2021. Shorter version [HSB22] of the same paper is presented in EMISA 2022. The third and the last published paper [SB21a] named *"From Conceptual Models to Knowledge Graphs: A Generic Model Transformation Platform"* was a collaboration of Smajevic and Bork and was published in 2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C) - Tools & Demonstrations Track.

CHAPTER 2

Background

In this chapter, we will go through different terms and topics used throughout the thesis. This includes an introduction to what Enterprise Architecture Management is, what is considered as graph analysis, what is quantitative graph theory, some information on graph visualizations, meta-models like Ecore, ArchiMate, ADOxx, model transformation, graph structure format GraphML, and goal question metric.

2.1 Enterprise Architecture Management

Enterprise Architecture Management (EAM) "*emerged as a way to deal with organizational complexity and change in an increasingly turbulent business environment*" [Ahl+12]. In the 1980s, system engineers tried to have a holistic perspective of the whole organization that would help them design the information system, but they realized that they could design good software if they had a better understanding of the organization [Ahl+12]. Ahlemann et al. define EAM as "*management practice that establishes, maintains and uses a coherent set of guidelines, architecture principles and governance regimes that provide direction for and practical help with the design and the development of an enterprise's architecture in order to achieve its vision and strategy*" [Ahl+12]. Historically, EAM has developed throughout the different phases as described in Fig. 2.1.

Zachman and his *Zachman Framework* [Zac87] heavily impacted the formation of EAM. At that time, Zachman realized that term *architecture* was used widely, but it had many different meanings. He introduced the framework that introduces different architecture perspectives, such as enterprise, technical, or system models. This was done by having different architecture descriptions like function, network, or data [Ahl+12].

In the next phase of EAM development, there were issues with spending on information systems and technology. This included solving local issues, introducing isolated systems, having many redundancies. Naturally, stakeholders felt that more transparent decision-

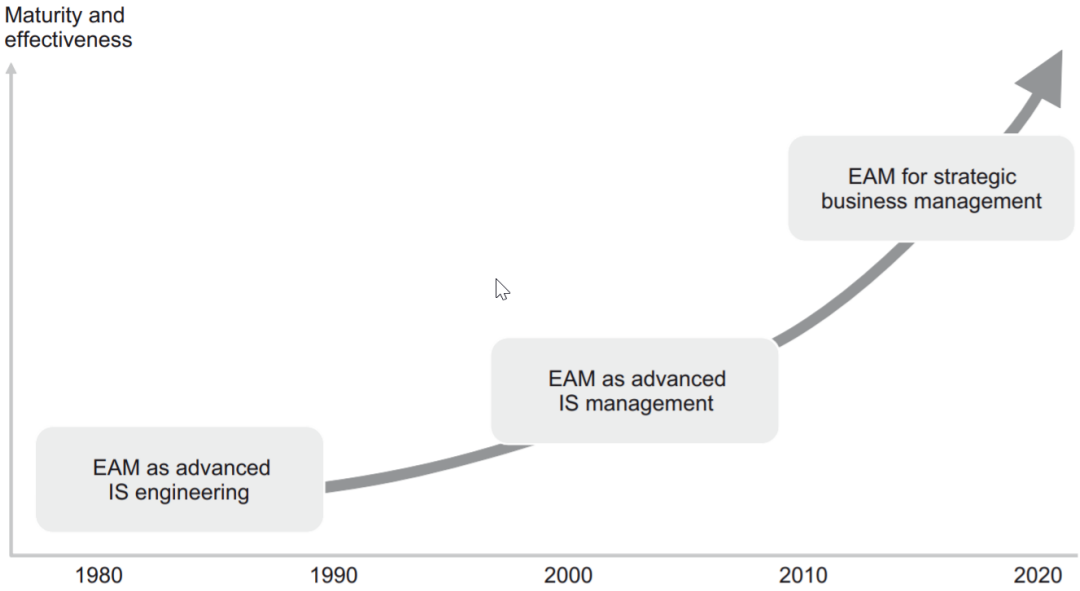


Figure 2.1: Development phases of EAM throughout the years [Ahl+12].

making was needed, and IT management processes and governance mechanisms had to be improved to achieve that. Based on this, new EAM frameworks were developed [Ahl+12]. One of the examples is The Open Group Architecture Framework (TOGAF).

Nowadays, EAM is not just a part of IT. It is a strategic function of a board member on the top level. This is because EAM can achieve full potential if it is linked directly to a business strategy [Ahl+12].

2.1.1 The Open Group Architecture Framework

Vicente, Gama, and Mira da Silva describe TOGAF as "a framework for developing an EA"[Gro18; VGM13]. From the development and maintenance perspective "it was developed and is currently maintained as a standard by The Open Group (TOG). The first version of TOGAF, in 1995, was based on the US Department of Defenses Technical Architecture Framework for Information Management (TAFIM)" [Gro18; VGM13]. *"Each version of the standard is developed collaboratively by the members of the TOG Architecture Forum"* [Gro18; VGM13]. *"The first seven versions of TOGAF addressed technology architecture based on its adoption in businesses at the time each was written. In 2002, Version 8 was published, which expanded the scope of TOGAF from a pure technology architecture to an EA, by including business and information systems architecture in the new version"* [Van09; VGM13]. Later in 2009, "TOGAF 9 was released with new features as a modular structure, a content framework specification, extended guidance, and additional detail" [Gro18; VGM13]. "TOGAF provides the methods and tools for assisting in the acceptance, production, use, and maintenance of an EA" [Gro18; VGM13]. "It is one of the leading architecture frameworks worldwide, and in its latest version, there is an increasing reflection on the use of the architecture and its governance" [Van09]. "It

is based on an iterative process model supported by best practices and a reusable set of existing architecture assets" [Gro18; VGM13]. "The TOGAF document focus on EA key concepts and TOGAF Architecture Development Method (ADM), a step-by-step approach to developing an EA" [JPT09; VGM13].

2.1.2 Conceptual Modelling

Conceptual modeling topic has had a growing interest over the last decade. As the interest in this topic grows, it leads to different views and opinions, which creates a ground for a debate that can lead to misunderstanding. Robinson et al. discuss this in [Rob+15], where they try to gather leading researchers to identify and discuss their views on conceptual modeling. They discuss the definition, purpose, and benefits of conceptual modeling. The following questions are covered [Rob+15]:

1. *"What is a conceptual model? (and how does the conceptual model relate to the real world (problem) and to the computer model?)"*
2. *"What is the purpose of conceptual modeling?"*
3. *"What are the benefits of a conceptual model?"*

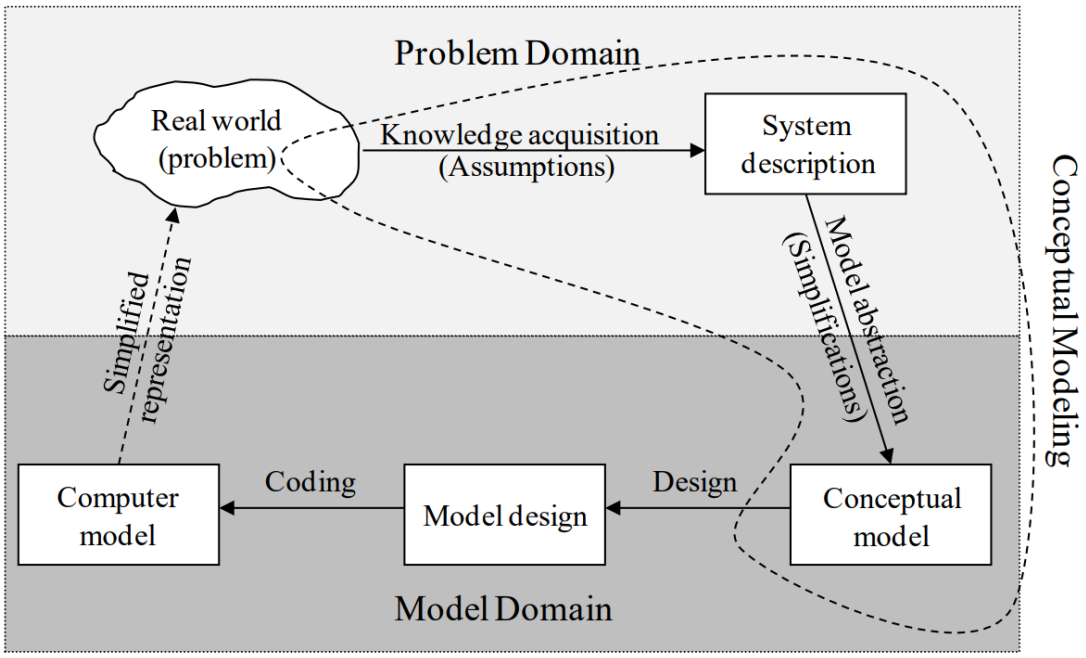


Figure 2.2: Artifacts of conceptual modeling by Robinson [Rob+15]

Below are the summarized views on topic *conceptual modeling* with the respect to definition, purpose, and benefits by different researches.

Definition of conceptual model [Rob+15]

Arbez and Birta: *"The conceptual model is a concise and precise consolidation of all goal-relevant structural and behavioral features of the SUI presented in a predefined format. "*
Robinson: *"A non-software specific description of the computer simulation model (that will be, is or has been developed), describing the objectives, inputs, outputs, content, assumptions and simplifications of the model. "*
Tolk: *"Result of the processes leading from the task to the specification of the conceptualization of the ontological structure of the problem domain, comprising assumptions and constraints on all relevant modeling decisions."*
Wagner: *"A solution independent description of a real world problem domain, from which a platform independent simulation design model can be derived for a given set of research questions."*

Purpose of conceptual modeling [Rob+15]

Arbez and Birta: *"The conceptual model should enable all stake holders to discuss the SUI's behavior and it must be sufficiently comprehensive to serve as a specification for a computer program."*
Robinson: *"A simulation model cannot exist without a conceptual model. A documented conceptual model is for communication."*
Tolk: *"Capturing and communicating the conceptualization with intended as well as potentially unforeseen simulation users."*
Wagner: *"To capture a sufficiently large, and sufficiently complete, part of the real world problem domain, for which a simulation study is to be performed, in such a way that all kinds of research questions concerning this domain can be investigated."*

Benefits of conceptual model [Rob+15]

Arbez and Birta: *"The conceptual model ensures that key SUI features (e.g. behavior, granularity) evolve from discussion with all stakeholders rather than from a programming bias."*
Robinson: *"A documented conceptual model is the basis for guiding all activities in simulation model development and use."*
Tolk: *"Building trust by unambiguously documenting the model – which is the foundation of the resulting simulation – which is pivotal in case of reuse or composition."*
Wagner: *"The CM can help to clarify questions about the scope and purpose of a simulation project, and it is an asset that can be re-used for making different solution designs for different research questions."*

We will follow the definition of Robinson where the artifacts of conceptual modeling are shown in Fig. 2.2.

2.1.3 Model Transformation

According to Langer et al., Model-Driven Engineering (MDE) "*places models as first-class artifacts throughout the software lifecycle. In this context, model transformations are crucial for the success of MDE, being comparable in role and importance to compilers for high-level programming languages, for bridging the gap between design and implementation*" [Lan+12].

In the past, several model transformation approaches were developed where "most of them are based on the abstract syntax of modeling languages which is defined by so-called metamodels" [Lan+12]. Regarding the metamodels, they "*describe by a limited set of UML class diagram concepts the object structure for computer internally representing and persisting models*" [Lan+12].

It is stated in the same paper that "modeling language engineering in MDE comprises at least two components" [Lan+12]. *"First, the abstract syntax of a language has to be defined by a metamodel, i.e., a model defining the grammar of the language. Second, to make a language more usable, a mapping of abstract syntax elements to concrete syntax elements (such as rectangles, edges, and labels) has to be provided"* [Lan+12].

A general explanation stated by the same paper is that in "transformation engineering in general, a model transformation takes a model as input and generates a model as output" [Lan+12].

2.2 Metamodels

2.2.1 Ecore

According to Eclipse, the *Eclipse Modeling Framework (EMF)* is a project which *"is a modeling framework and code generation facility for building tools and other applications based on a structured data model"* [Ecl21a]. The model specification is described in XMI which can be used "to produce a set of Java classes for the model" [Ecl21a] which means it is facility to build up code based on model specification.

The framework itself consists of three parts: *EMF*, *EMF.Edit* [Ecl21a], and *EMF.Edit*. The core of the *EMF* is the meta model called *Ecore*. This model allows creating other models. The hierarchy of the components used to create this meta model can be seen in Fig. 2.3. Relevant concepts of the Ecore meta model can be seen in Fig. 2.4.

2.2.2 ArchiMate

"*The ArchiMate EA modeling language was developed to provide a uniform representation for architecture descriptions. It offers an integrated architectural approach that describes and visualizes the different architecture domains and their underlying relationships and dependencies*" [JPT09; VGM13]. Furthermore, the goal of the ArchiMate project "is to provide domain integration through an architecture language and visualization techniques

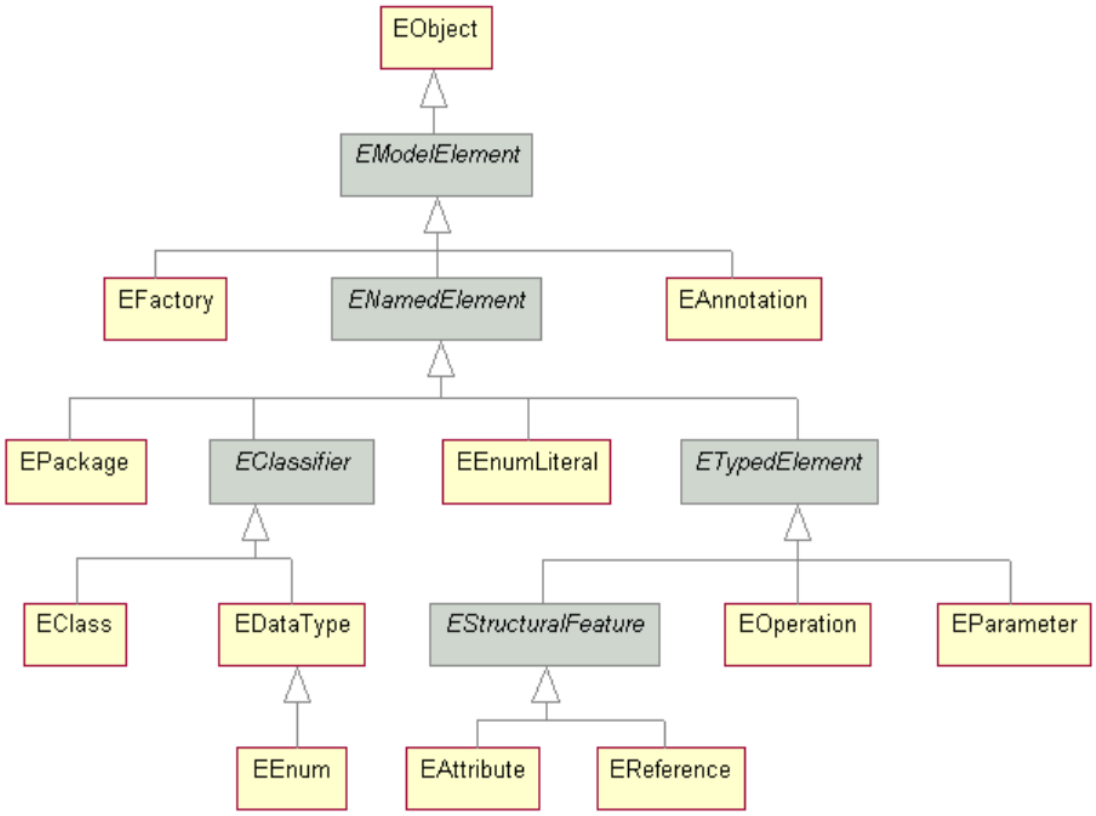


Figure 2.3: Ecore components hierarchy [Ecl21b].

that picture these domains and their relations, providing the architect with instruments that support and improve the architecture process" [VGM13]. Additionally "in a short time, ArchiMate has become the open standard for architecture modeling in the Netherlands; it is now also becoming well known in the international EA community" [JPT09].

Nowadays, ArchiMate is considered as popular EA Modeling language [Lan+09; OMG19]. ArchiMate represents the layered view of an organization. Many EAM tools provide the ArchiMate model by default or try to enrich it to increase the model value.

Regarding the ArchiMate structure "*the domains of business, application, and infrastructure are connected by a service orientation paradigm, where each layer exposes functionality in the form of a service to the layer above*" [VGM13; Gro12]. Additionally ArchiMate, "also distinguishes between active structure, behavior, and passive structure elements, having also another distinction between internal and external system view" [VGM13; Gro12] and "ArchiMate is a formal visual design language, supports different viewpoints for selected stakeholders, and is flexible enough to be easily extended" [VGM13; Gro12]. On the other side, ArchiMate has limited semantic distinction [PB17] and also provides limited information processing contained in the model [BJS13].

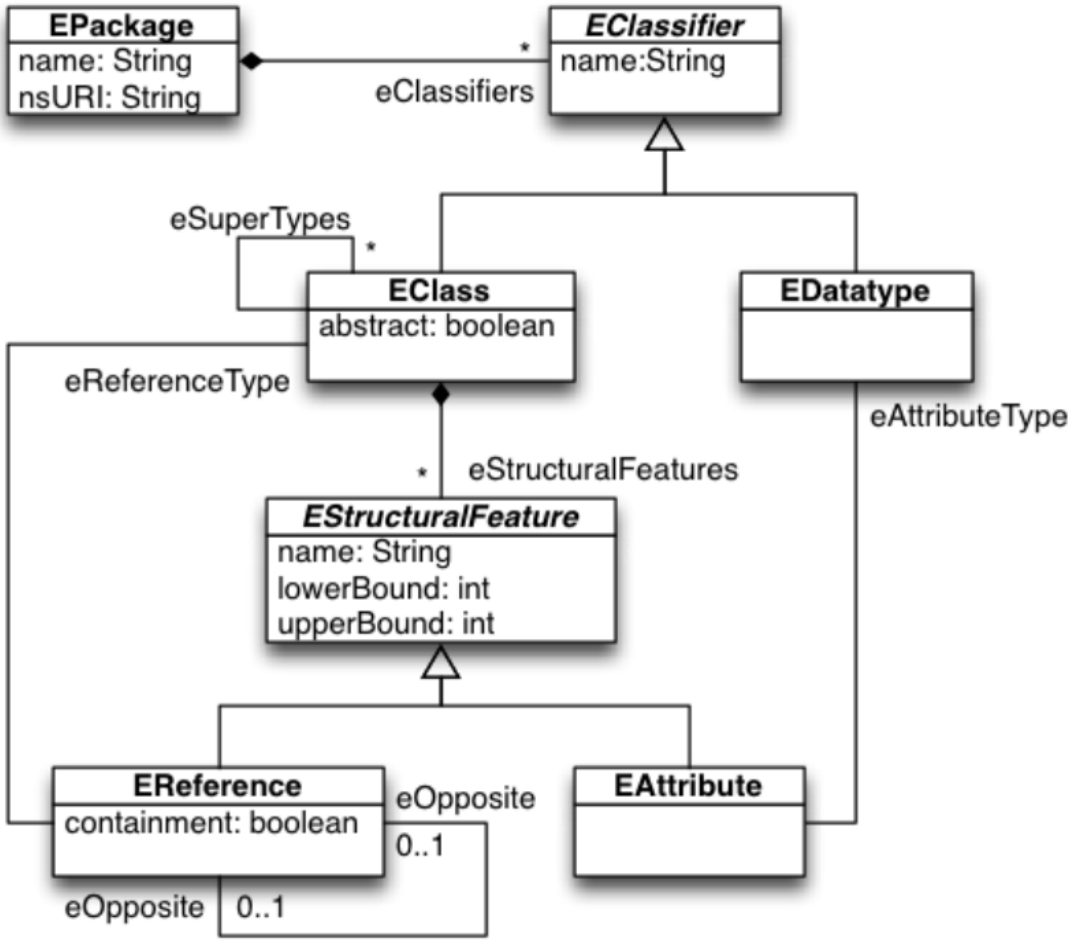


Figure 2.4: Most relevant concepts and their relations in Ecore [ecl21].

2.2.3 ADOxx

Based on its website, *ADOxx "is the meta-modeling development and configuration platform for implementing modeling methods"* [ADO21b]. Furthermore, "implementation of full-fletched modelling methods can be realized using the platform, consisting not only of a modelling language, but also of modelling procedure and the corresponding functionality in the form of mechanisms and algorithms." [ADO21b]. All developed artifacts are contained in *Application Library* which contains definition of model types, classes, and relation classes (this includes both: abstract and concrete level) [ADO21b]. This can be seen in Fig. 2.7.

Fill and Karagiannis give more details about ADOxx such as that in the background, ADOxx meta^2 model is developed in C++ [FK15]. *"User-specific meta-models are derived from classes of the ADOxx Meta Model and described by a meta modeler using the proprietary ADOxx Library Language (ALL)"* [FK15]. ADOxx can export models in

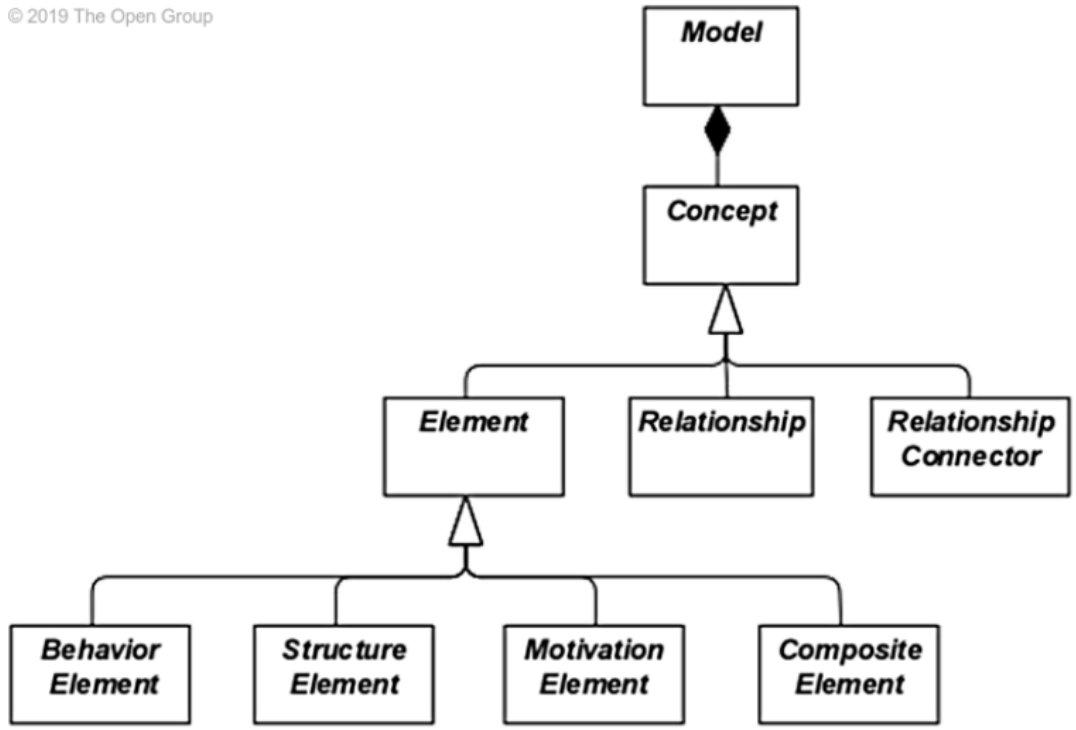


Figure 2.5: Top-Level Hierarchy of ArchiMate Concepts [Gro21].

ADL or XML format [FK15]. Meta model of ADOxx can be seen in Fig. 2.8.

Core part of ADOxx consists of classes and relation classes which are grouped by model types [FK15]. In both cases, these classes can have attributes that can be help text or regular expressions [FK15].

2.3 Graph Analysis

Pachayappan and Venkatesakumar explain that a graph "*connects two or more entities where entities can be anything: human beings, machines, animals, characters in a movie, and variables in literature*" [PV18]. Furthermore, the paper states that "in Graph Theory, these entities are considered as *Nodes* while the relationships are considered as *Edges* and their connections form a graph"" [PV18]. There are three types of graphs *undirected*, *directed*, and *mixed* graphs. Analyzing a graph is a wide area and many applications on how to analyze a graph exists. We will briefly introduce *quantitative graph analysis* and *visual analysis* in the following sections.

2.3.1 Quantitative Graph Theory

Dehmer et al. define *Quantitative Graph Theory* as "*a measurement approach to quantify structural information of networks*" [Deh+14]. It is a branch of Graph theory and it is

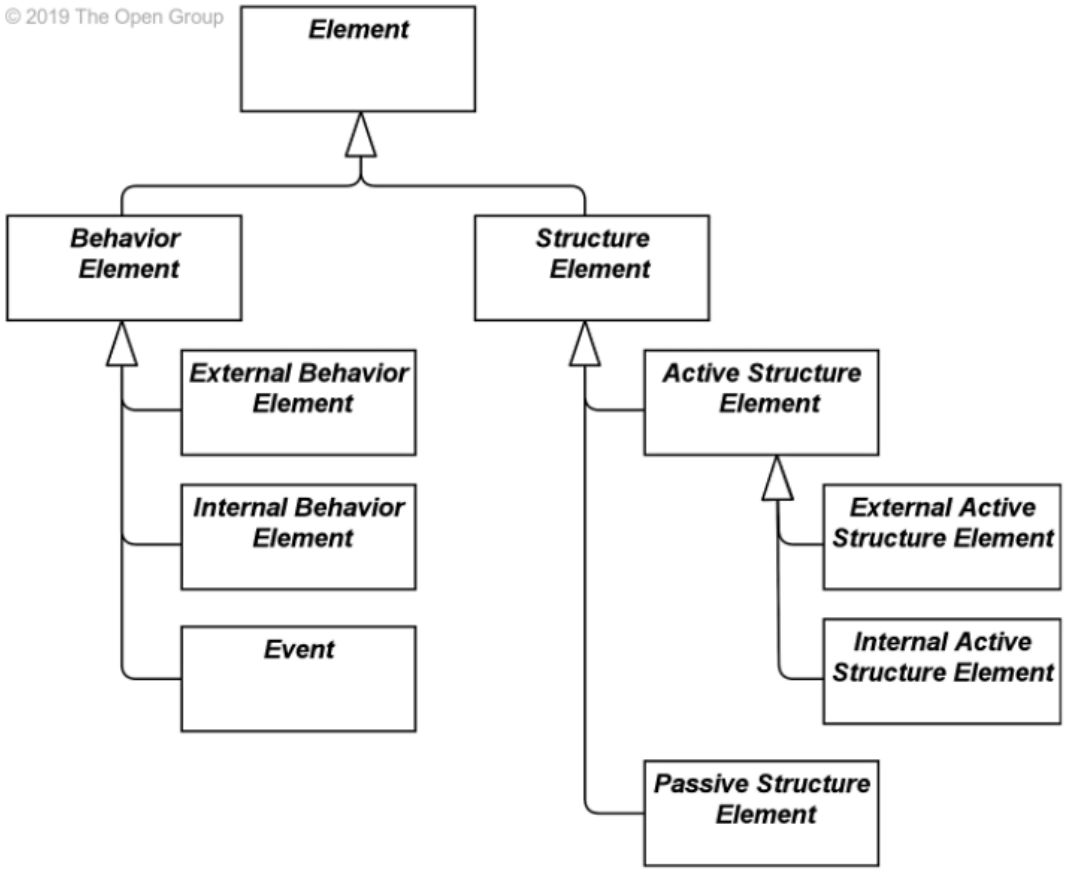


Figure 2.6: Hierarchy of ArchiMate Behavior and Structure Elements [Gro21].

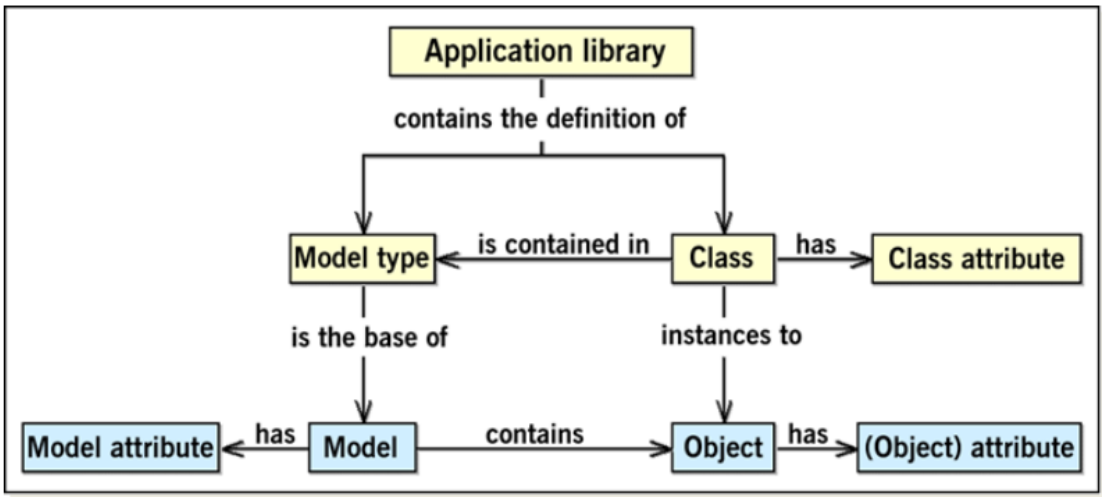


Figure 2.7: The ADOxx library definition [ADO21b]

concerned about quantification of structural information in networks [DES17]. Compared to classical graph theory, Dehmer, Emmert-Streib, and Shi describe "some highlights of

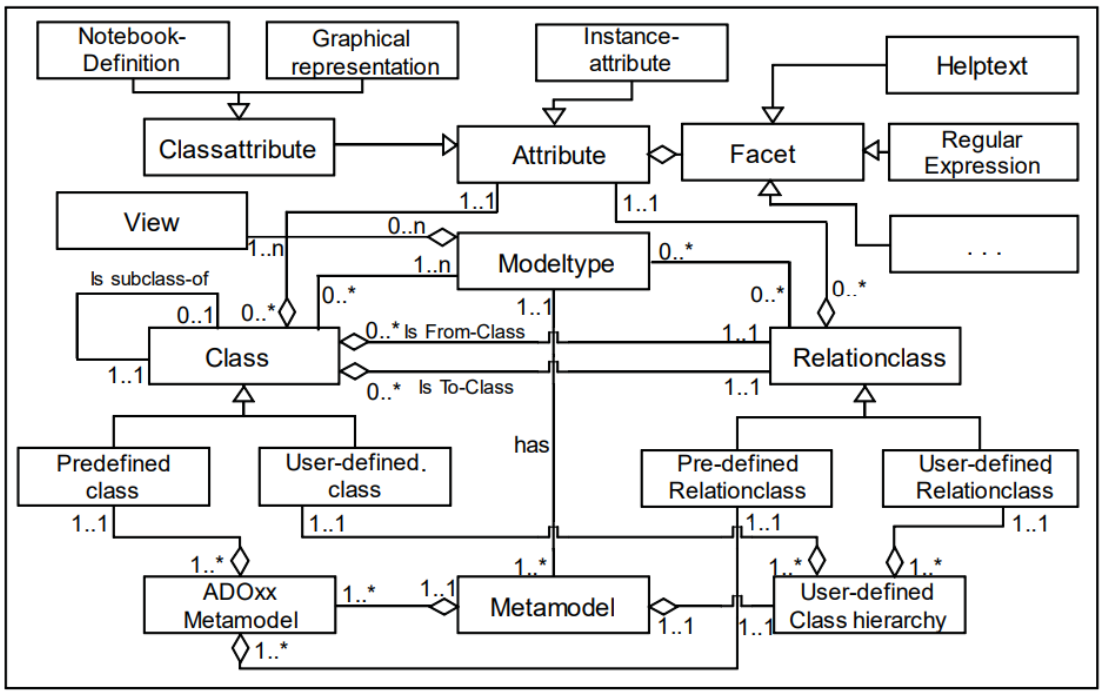


Figure 2.8: The ADOxx meta^2 model [FK15].

the new branch of quantitative graph theory and explains its significant different features compared to classical graph theory" [DES17]. Furthermore, authors state that the main goal of quantitative graph theory "is the structural quantification of information contained in complex networks by employing a measurement approach based on numerical invariants and comparisons" [DES17]. Moreover, "the methods as well as the networks do not need to be deterministic but can be statistic" [DES17]. By doing this authors emphasize that "this complements the field of classical graph theory, which is descriptive and deterministic in nature"[DES17]. The same paper provides examples "of how quantitative graph theory can be used for novel applications in the context of the overarching concept of network science" [DES17].

Methods used in quantitative graph theory described in [DES17] are as follows:

Comparative graph analysis. "Comparative graph analysis means determining the structural similarity or distance between two or more networks." [DES17]

Graph characterization. "In quantitative graph theory, graph characterization relates to determining the complexity of a given network by using numerical graph invariants. Graph invariants are graph measures to characterize graphs structurally which are invariant under isomorphism." [DES17]

Software for quantitative graph analysis. "Programs developed for the statistical programming language R" [DES17]

2.3.2 Graph Visualisation

Schulz and Schumann emphasize that "visualizing structures is a hot topic in the domain of information visualization" [SS06]. Furthermore, authors state that "visualization of graphs has proven to be very useful for exploring structures in different application domains" [SS06]. Graph visualisation can be understood differently. Two terms that should be distinguished are *graph drawing* ("which focuses on optimized layouts for modeling representations of networks" [SS06]) and *information visualization* ("which focuses on hierarchies focusing on very large structures, different views, and interactivity") [SS06].

Additionally, authors give "a systematic view of the problem of graph visualization by combining both approaches" [SS06] as follows:

Hierarchy Representations. "There are two principal alternatives to classify visualization techniques for hierarchies" [SS06]:

- explicit vs. implicit
- axes-oriented vs. radial

Network Representations. "While the organization of data in a hierarchical manner is a powerful pattern used frequently throughout many fields of application, hierarchies and especially trees are just a tiny subclass compared to the huge graph class of networks." [SS06]

Additionally one should look for constraints on network visualization [SS06]:

- "User Constraints: compatibility of visualization and user, compatibility of visualization and interaction techniques, compatibility of different visualization techniques." [SS06]
- "Data Constraints: static data vs. dynamic updates, size, and density of the graph, graph topology." [SS06]

2.3.3 GraphML

Brandes et al. talk about the general requirement for graph structure format specification and introduce GraphML as follows: "*Among the multitude of software packages that process graphs, some are dedicated graph packages while others operate on graph structures implicitly. All of them have in common the need to input existing data and to output their computation results in files or streams. GraphML (Graph Markup Language) is an XML-based format for the description of graph structures, designed to improve tool interoperability and reduce communication overhead*" [Bra+02]. The same paper states that GraphML "it is open to user-defined extensions for application-specific data. Thanks to its XML syntax, GraphML-aware applications can take advantage of a growing number of XML-related technologies and tools, such as parsers and validators" [Bra+02].

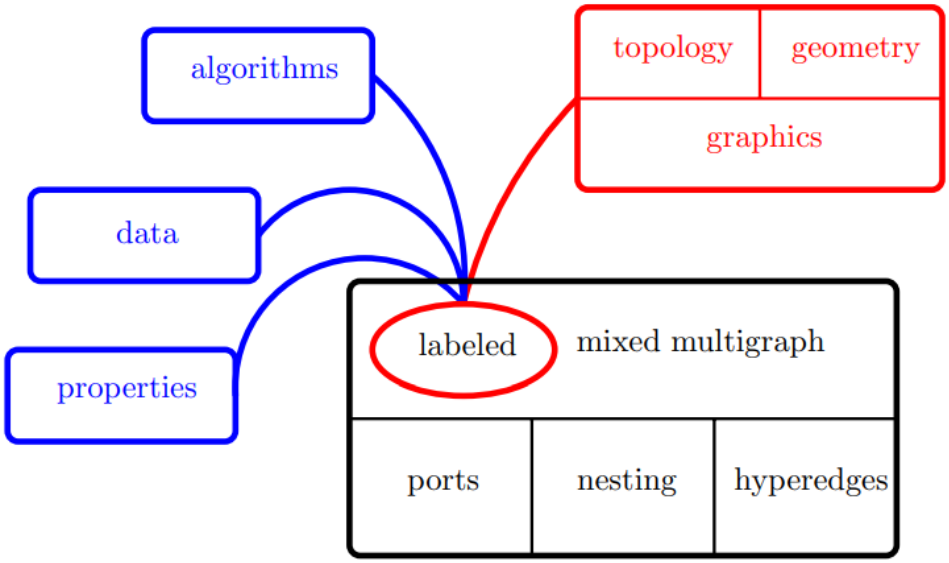
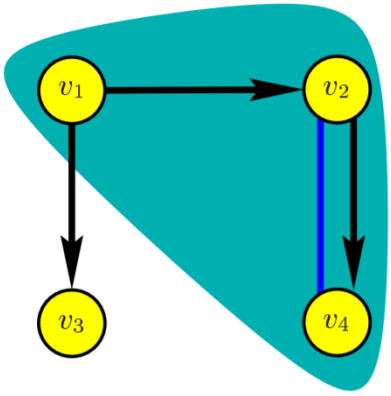


Figure 2.9: The basic graph model of GraphML [Bra+02].

Brandes and Pich provide more information about GraphML with a "key feature of GraphML is the separation into structural and data layer, both conceptually and syntactically; this enables applications to extend the standard GraphML vocabulary by custom data labels that are transparent to other applications not aware of the extension. Furthermore, applications are free to ignore unknown concepts appearing in the structural layer, such as *<port>s*, *<hyperedge>s* or nested *<graph>*s" [BP05].

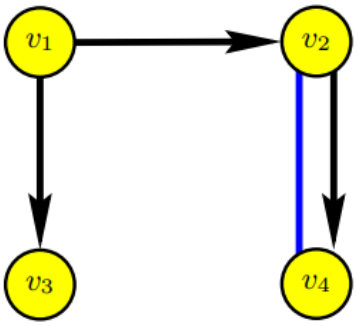



```
<graph edgedefault="directed">
  <desc>Mixed multigraph with a hyperedge</desc>

  <node id="v1"/>
  <node id="v2"/>
  <node id="v3"/>
  <node id="v4"/>

  <edge source="v1" target="v2"/>
  <edge source="v1" target="v3"/>
  <edge source="v2" target="v4"/>
  <edge source="v2" target="v4" directed="false"/>

  <hyperedge>
    <endpoint node="v1" type="out"/>
    <endpoint node="v2" type="in"/>
    <endpoint node="v4"/>
  </hyperedge>
</graph>
```

Figure 2.10: A hyperedge to v1, v2, and v4, where v1 is a source, and v2 is a sink. [Bra+02].

Brandes and Pich add more information on benefits of using GraphML: *"Thanks to its XML syntax, GraphML can be used in combination with other XML based formats:*

On the one hand, its own extension mechanism allows to attach <data> *labels with complex content (possibly required to comply with other XML content models) to GraphML elements, such as Scalable Vector Graphics describing the appearance of the nodes and edges in a drawing; on the other hand, GraphML can be integrated into other applications, e.g. in SOAP messages"* [BP05].




```
<graph edgedefault="directed">
  <desc>This is a mixed multigraph</desc>

  <node id="v1"/>
  <node id="v2"/>
  <node id="v3"/>
  <node id="v4"/>

  <edge source="v1" target="v2"/>
  <edge source="v1" target="v3"/>
  <edge source="v2" target="v4"/>
  <edge source="v2" target="v4" directed="false"/>
</graph>
```

Figure 2.11: The graph on the left represented in the most basic layer of GraphML. [Bra+02].

GraphML consists of the following XML constructs [BP05]:

Mixed Multigraphs. "A mixed multigraph is a graph that may contain both directed and undirected edges and may have loops and multi-edges. The representation chosen for GraphML is a simple list of nodes and edges. GraphML defines XML tags *<graph>*, *<node>*, and *<edge>* for this purpose." [Bra+02].

Ports. "A port is a subset of the incidence relations of a node and can be viewed as a part of a node to which edges may attach. In electrical circuits for instance ports can be legs of a chip, and in graph drawing, they may be used to specify points at which edges connect to a node. In GraphML *<port>s* appear as nested subelements of *<node>s*, and *<edge>s* may specify *<port>s* they attach to for both of their endpoints" [Bra+02].

Hyperedge. "A hyperedge is a subset of nodes, together with a classification of these nodes into inputs, outputs, or neither of the two. A GraphML tag *<hyperedge>* may therefore contain any number of *<endpoint>s* which, in turn, refer to *<node>s*, but also classify these nodes using the XML attribute type." [Bra+02].

Nested Graphs. "A nested graph is a graph occurring in an element of another graph. There are many models of hierarchical graphs, e.g., allowing more than one nested graph per element or a graph to be contained in more than one element. Each item of a graph, i.e. each *<node>*, *<edge>*, or *<hyperedge>* may contain one nested *<graph>* element. Though simple, this model is sufficiently general to support all of the above variants. More than one contained graph can be expressed by defining

a single contained graph that has a node for each of the child elements, and a contained graph appearing in different places can be referenced using a *<locator>* element." [Bra+02].

2.4 Goal Question Metric

Solingen et al. describe *The Goal Question Metric (GQM) approach "is based upon the assumption that for an organization to measure in a purposeful way it must first specify the goals for itself and its projects, then it must trace those goals to the data that are intended to define those goals operationally, and finally provide a framework for interpreting the data with respect to the stated goals"* [Sol+02]. Furthermore, authors emphasize that "it is important to make clear, at least in general terms, what informational needs the organization has so that these needs for information can be quantified whenever possible, and the quantified information can be analyzed a to whether or not the goals are achieved" [Sol+02].

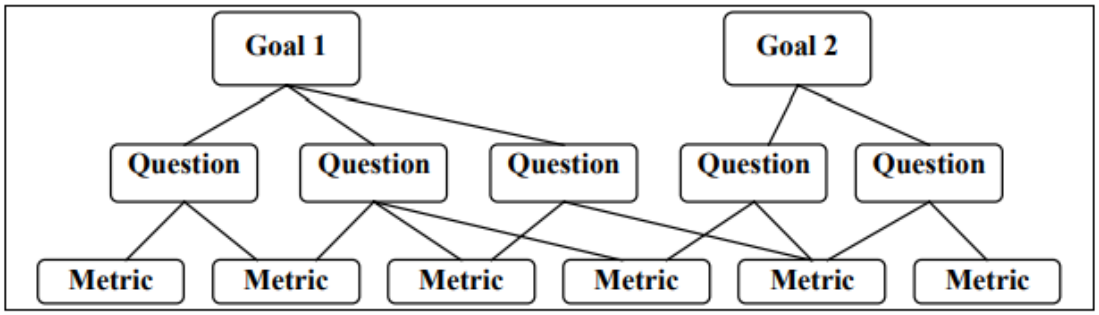


Figure 2.12: GQM model is a hierarchical structure [Sol+02].

From the historic perspective, the approach "was originally defined for evaluating defects for a set of projects in the NASA Goddard Space Flight Center environment" [Sol+02]. "The application involved a set of case study experiments" [BW84] "and was expanded to include various types of experimental approaches" [Sol+02]. "Although the approach was originally used to define and evaluate goals for a particular project in a particular environment, its use has been expanded to a larger context" [Sol+02]. "It is used as the goal-setting step in an evolutionary quality improvement paradigm tailored for a software development organization, the Quality Improvement Paradigm, within an organizational framework, the Experience Factory, dedicated to building software competencies and supplying them to projects" [Sol+02].

Solingen et al. state that "the result of the application of the Goal Question Metric approach application is the specification of a measurement system targeting a particular set of issues and a set of rules for the interpretation of the measurement data" [Sol+02]. The resulting measurement model has three levels [Sol+02]:

Conceptual level (GOAL). "A goal is defined for an object, for a variety of reasons,

Goal	Purpose Issue Object (process) Viewpoint	Improve the timeliness of change request processing from the project manager's viewpoint
Question		What is the current change request processing speed?
Metrics		Average cycle time Standard deviation % cases outside of the upper limit
Question		Is the performance of the process improving?
Metrics		$\frac{\text{Current average cycle time}}{\text{Baseline average cycle time}} * 100$ Subjective rating of manager's satisfaction

Figure 2.13: The complete Goal/Question/Metric Model [Sol+02].

with respect to various models of quality, from various points of view, relative to a particular environment." [Sol+02]. Objects of measurement are [Sol+02]:

- Products: Artifacts, deliverables, and documents that are produced during the system life cycle; E.g., specifications, designs, programs, test suites.
- Processes: Software-related activities normally associated with time; E.g., specifying, designing, testing, interviewing.
- Resources: Items used by processes in order to produce their outputs; E.g., personnel, hardware, software, office space.[Sol+02]

Operational level (QUESTION). "A set of questions is used to characterize the way the assessment/achievement of a specific goal is going to be performed based on some characterizing model. Questions try to characterize the object of measurement (product, process, resource) with respect to a selected quality issue and to determine its quality from the selected viewpoint." [Sol+02]

Quantitative level (METRIC). "A set of data is associated with every question in order to answer it in a quantitative way." [Sol+02] The data can be:

- Objective: If they depend only on the object that is being measured and not on the viewpoint from which they are taken; E.g., number of versions of a document, staff hours spent on a task, size of a program.
- Subjective: If they depend on both the object that is being measured and the viewpoint from which they are taken; E.g., readability of a text, level of user satisfaction.[Sol+02]

2.5 Smell Detection

2.5.1 EA Smell Detection

Hacks et al. in the paper [Hac+19] propose to combine the concept of Technical Debt [Cun92] by Cunningham with the concept of Enterprise Architecture to a so-called *EA Debts*. This was not to cover technical aspects but instead, EA debts were to provide a more holistic view of the entire organization, including flaws in the organization structure. This proposal lacked an effective way to measure EA Debts. Furthermore,Salentin and Hacks, in a new paper [SH20b] used the concept of Code Smells to adopt it and build EA Smells since it was more popular to measure Technical Smells. They took into consideration 56 Code smells and made a catalog of 45 EA Smells [Sal+21]. Later, this was extended by Lehmann et al. in [Leh+20] by taking the inspiration from process anti-pattern, and Tieu and Hacks in their paper [TH21] where they were inspired by software architecture smells.

2.5.2 Code Smell Detection

Initial work on code smell detection goes back to Fowler in his book [Fow99], where a work on code refactorings was published. Later this was developed further to code smells with respect to UML Class Diagrams by Arendt and Taentzer; Suryanarayana, Samarthyam, and Sharma; Haendler; Mumtaz et al. in [AT10; SSS14; Hae18; Mum+19] respectively. Some of the code smells can be seen in Figure 2.14.

Violated Design Princ.	Software design smell based on (Fowler et al., 1999; Suryanarayana et al., 2014)	Aliases (used in research or industry) and smells with similar symptoms	Symptoms description based on (Fowler et al., 1999; Suryanarayana et al., 2014)	DECOR	JDeodorant	EIMF Refactor	Smell false positives oriented to (Fontana et al., 2016)
Abstraction	DataClump	(a kind of) MissingAbstraction	Clumps of data used instead of a unit (e.g., class)	–	–	√	–
	MultifacedAbstraction	LargeClass, GodClass, lack of cohesion	Unit (e.g., classes) with more than one responsibility	√*	√*	√*	State DP, generic class, e.g., configuration class, GUI widget toolkits
	UnutilizedAbstraction	UnusedClass, SpeculativeGenerality	Not or barely used units (e.g., class or method)	√	–	√	recently developed program elements not yet covered by tests, null implementation
	DuplicateAbstraction	CodeClone, DuplicatedCode, functionally similar methods (as kind of DuplicateAbstraction)	Multiple units (classes or methods) with identical (or similar) internal and/or external structure or behavior	–	√*	√*	inherited or overridden method
Encaps.	DeficientEncapsulation	Hideable public attributes or methods	The accessibility of attributes or methods is more permissive than actually required	–	–	–	–
	LeakyEncapsulation	–	A unit that exposes implementation details via its public interface	–	–	–	–
Hierarchy	SpeculativeHierarchy	SpeculativeGenerality, speculative general types	One or more types in a hierarchy are used speculatively (only based on imagined needs)	√	–	√	–
	UnnecessaryHierarchy	Taxomania (taxonomy mania)	A variation between classes is mainly/only captured in terms of data (structural features)	–	–	–	–
	DeepHierarchy	DistortedHierarchy	An unnecessarily deep hierarchy	–	–	–	–
	MultipathHierarchy	RepeatedInheritance, DiamondInheritance	A subtype inherits both directly and indirectly from a supertype	–	–	√	–
Modularization	FeatureEnvy	(a kind of) BrokenModularization, Misplaced operations	Methods are more interested in features owned by foreign classes than in features of the owning class	–	√	–	Visitor DP, Strategy DP, Decorator DP, Proxy DP, Adapter DP
	DataClass	(a kind of) BrokenModularization, RecordClass, DataContainer	Classes providing data but having no (or only few) methods for operating on them	√	–	–	ExceptionHandlingClass, LoggerClass, SerializableClass, configuration class, Data Transfer Object (DTO)
	CyclicDependendModularization	CyclicDependency, (Dependency) Cycles	Two or more units (e.g., classes, methods) mutually depend on each other	–	–	–	Visitor DP, Observer DP, AbstractFactory DP
	MessageChain	(a kind of) BrokenModularization	A client unit (e.g., method) calls another unit, which then in turn calls another unit, and so on (navigation through class structure)	√	–	–	Builder DP, Facade DP, test-class method

Figure 2.14: Overview of 14 software design smells categorized by design principles [Hae18]

CHAPTER 3

Related work

3.1 EA and Graph-based Analysis

We have already introduced some of the related work in Sec. 1.1.1. Here we now expand this topic with the related work in this domain. As already stated, there is a lot of research to give more options to enterprise architects with additional analysis techniques [BMS09]. Buckl, Matthes, and Schweda emphasize the necessity of automated tools that will provide different analysis techniques. Additionally they put the emphasis on scalability of the approaches in order to handle large EA models. In their work, Lantow et al. state that *"Modern analysis approaches should combine interactive visualizations with automated analysis techniques"* [Lan+16]. Additionally, Santana et al. emphasise the need to have a proper tool which will enable EA analysis. Barbosa et al. were concerned on what EA analysis is, and therefore developed a taxonomy for EA analysis research [Bar+19]. This taxonomy consists of four dimensions: *EA Scope*, *Analysis Concern*, *Analysis Technique*, and *Modelling Language*. The last dimension was then separated in nine groups. Out of these nine groups, two are interesting for this thesis *ArchiMate-based* and *Graphs*. Moreover, different analysis techniques like *Metric-based*, *Structural analysis*, and *Visual analysis* are introduced.

We now move into research on the intersection of EA analysis and graph-based analysis. Garg, Kazman, and Chen in their study [GKC06], propose a 3-tier architecture in order to enable the definition of different enterprise applications by transforming them into the graph structure. The reason for this representation is the tendency to provide better visual analysis through a "system of systems" view, reliability and redundancy via architectural analysis, and risk analysis. This proposal does not contain modeling possibility and is technical and conceptual by nature. Aier presents a tool called *EA Builder* [Aie06]. The purpose of this tool is to identify clusters in the candidates for service in a service-oriented architecture which are presented in graph structures. EA Builder provides the possibility to model the EA via *Event-Driven Process Chains*. Unfortunately,

this tool is not available anymore. Similar to this, Iacob and Jonkers present the idea to quantitatively analyze "layered, service-oriented EA models, which consists of top-down propagation of workload parameters and bottom-up propagation of performance and cost measures" [IJ06].

Another approach is focused on using Bayesian Belief Networks. This was proposed by Holschke et al. in their study [Hol+09] where they use these networks to implement failure impact analysis on models developed in ArchiMate. The work presented in [San+16] proposes a *cognitive-structural diagnosis analysis method* that combines manual inspection by experts (i.e., enterprise architects) with the computational power of graph analysis algorithms. In contrast to our approach, their approach is data-driven and manual, i.e., not using a modeling language and is not following a model-driven approach. The authors claim their approach would *"benefit considerably from a software plugin capable of converting data from architecture models to networks."* [San+16, p. 5]

Santana, Fischbach, and Moura perform a broad literature review of EA network analysis in their study [SFM16]. They identify applied measures and main achievements. In total, 29 papers were chosen and analyzed. The concepts from EA domains from TOGAF are used to classify all the measures. They found that 22 out of 29 papers have application architecture as the input. 10 out of 29 papers contained only an illustrated application, 5 out of 29 proposed measures theoretically. Furthermore, "most of the measures (51 out of 67) are designed for relationship level". Summary of the network analysis initiatives can be seen in Fig. 3.1 and Fig. 3.1. One of the recommendations for future research is the tool development for EA analysis. It is mentioned here that [LH12] presents the transformation of the BPMN model to network structure suitable for analysis while [Ram+14] presents additional example of tool support.

We now briefly discuss the EA analysis approaches that do not involve graph structures. Österlind, Lagerström, and Rosell in their study [ÖLR12] propose a tool named *Enterprise Architecture Analysis Tool* (EAAT). The idea behind this tool is to extend ArchiMate concepts "with variables that are computed for structurally analyzing the EA" [ÖLR12]. These variables are then used to calculate the metrics for *Coupling*, *Complexity*, *Size*, and *Modifiability* which should help enterprise architects in their decisions on modifying the EA. More on the metrics side is proposed by Singh and Sinderen where they introduce "seven metrics to measure the criticality and impact of any element in an EA model" [SS15]. Another idea is to adapt ArchiMate modeling language itself to perform some analysis. This was done by Buschle, Johnson, and Shahzad where they present a tool that can be used to analyze the availability of EA components by using "fault tree formalism" [BJS13].

Antunes, Caetano, and Borbinha address the complexity of the models, which will have to be analyzed manually [ACB14]. The paper states the need for the appearance of automated analysis proposals. Furthermore, the paper discusses the use of ontologies for EA analysis "focusing on the particular case of description logics" [ACB14]. For that purpose, a "survey on analysis approaches is performed and the reasoning features provided by description logics are matched to the different types of EA analysis" [ACB14]. With the results there, "a set of constructs is proposed for the representation, integration,

Name of initiative	EA domain targeted[a]							Structural level[b]				Val	Primary Study
	GE	*HO*	*STK*	*BA*	*DA*	*AA*	*IA*	*Co*	*Rel*	*Mod*	*Net*		
Modularity	X								X			T	[18]
Number of components	X							X				T	[18]
Degree of connectedness	X										X	T	[18]
Number of types of relations	X								X			T	[18]
Intensity of the relations (weighted relation)	X								X			T	[18]
Multiplier effect in the flows of interactions	X								X			T	[18]
G&G metric	X									X		I	[28]
Service time actual (STA)		X		X	X	X	X	X	X			I	[30]
Purity of the domain				X		X			X			E	[39]
Purity of categories						X				X		E	[39]
Data sovereignty				X	X	X		X	X			E	[39]
Correct category dependencies						X			X			E	[39]
Distance of application landscapes		X		X	X	X			X			E	[39]
Clustering algorithms and centrality measures from Netdraw tool			X	X				X	X	X		I	[24]
Centrality measures (betweenness and degree centralities)			X	X			X		X			E	[34]
Clustering index						X	X				X	E	[34]
Average path length						X	X				X	E	[34]
Girvan/Newman's clustering algorithm				X		X				X		I	[36]
McCabe's cyclomatic number				X					X			I	[35]
Activity/passivity of a business process				X					X			I	[35]
Business-application alignment				X		X			X			I	[23]
Static alignment measure (SAM)				X	X				X			I	[27]
Degree of informational dependence					X	X			X			T	[26]
Degree of functional dependence						X			X			T	[26]
Cohesion (operationalized by embeddedness)						X			X			E	[12]
Coupling (operationalized by closeness centrality)						X			X			E	[12]
Structural complexity and procedural complexity						X		X				E	[12]
Data complexity						X			X			E	[12]
Connectivity strength weighted parameter frequency (CSWPF)						X		X	X			E	[38]
"The hidden structure" method						X			X			E	[7]
"The hidden structure" method		X		X	X	X	X		X	X		E	[10]
Chidamber and Kemerer's measure of coupling operationalized by DFI/DFO		X		X	X	X	X		X	X		E	[10]
Closeness centrality		X		X	X	X	X		X			E	[10]
Business information criticality of an application (BIO)					X	X		X	X			T	[25]
Architectural control points (ACP)						X			X			I	[11]
Fitness difference between architectural control point sets						X			X			I	[11]
Small world coefficient difference between initial and final stages						X					X	I	[11]
Absolute criticality of the a services S (ACS)						X			X			T	[29]
Average service depth (ASD)						X			X			T	[29]
POU – Provided operation utilization of a service						X				X		T	[29]
ROU – Required operations utilization of a service						X				X		T	[29]
Change cost of a service						X				X		T	[29]
Instability of element – IOE (e)						X			X			T	[29]
Whitney index (WI)						X				X		E	[37]
Change cost (CC)						X			X		X	E	[37]

Figure 3.1: Network analysis initiatives found in primary studies part 1. [SFM16].

Name of initiative	GE	HO	STK	BA	DA	AA	IA	Co	Rel	Mod	Net	Val?	Primary Study
Centrality measures - degree, closeness and betweenness						X			X			E	[37]
Architectural relevance						X			X			E	[8]
Cross-functional application measure (operationalized by closeness centrality)						X			X			E	[8]
Degree centrality for application modifiability, risk and cost estimation						X			X			E	[8]
Clustering algorithm (not detailed) for group applications according to their relations						X					X	E	[8]
Modularity – operationalized by global modularity and clustering coefficient						X					X	E	[8]
Betweenness centrality for identify critical points in terms of risk and operational value of an application						X			X			E	[8]
Eigenvector centrality for identifying critical applications in terms of failure						X			X			E	[8]
Landscape size (number of business related IT applications)						X			X			T	[33]
Interdependency between applications						X			X			T	[33]
Redundancy – different instances of major applications for business units				X		X			X			T	[33]
Infrastructure landscape size (number of physical servers/clients)							X		X			T	[33]
Impact analysis of a component in its neighborhood	X								X			I	[31]
Number of interfaces of the application to other applications						X			X			E	[21]
Business coverage redundancy/overlap				X		X			X			E	[21]
Number of processes covered by the application (scope of the application)				X		X			X			E	[21]
Discriminant functions several network measures				X					X			E	[40]
Path dependence based network model, hub centrality, net. connectedness						X			X			E	[9]
Degree, betweenness centrality calculated with weighted qualified edges						X			X			E	[32]
Process Page Rank algorithm				X				X	X			I	[41]
"The hidden structure" method						X			X			E	[13]
Process Clustering (Hierarchical multidimensional scaling method – MDS)				X					X	X		E	[42]

[a] EA domain targeted: GE – general EA domains; HO – holistic enterprise architecture; STK – stakeholder domain; BA – business architecture; AA – application architecture; DA – data architecture; IA – infrastructure archit.

[b] Structural level on which the measure was applied: CO – component; Rel – relation; Mod – module; Net – network.

Figure 3.2: Network analysis initiatives found in primary studies part 2. [SFM16].

and analysis of EA models" [ACB14]. As an example, "a demonstration of such constructs applied in compliance analysis is performed using an example scenario" [ACB14]. They propose "architecture that makes possible the usage of reasoning for performing analysis of the models, providing the required information for stakeholders" [ACB14]. On the other side authors mention a limitation that "different types of analysis rely on different types of techniques that offer features that cannot be offered by ontologies" [ACB14]. The additional limitation which is mentioned is "not to the ontology technology, but to the analysis itself, which is that the quality of the analysis is dependent on the quality of the information captured in the EA" [ACB14].

Furthermore, Antunes et al. add more to ontology-based enterprise architecture model analysis in another study [Ant+14]. This paper describes an application of ontology engineering to enterprise architecture. As a contribution it provides "an extensible architecture description language that includes an upper ontology that can be integrated with multiple domain-specific ontologies, each focusing on different concerns" [Ant+14], where "resulting integrated models can be automatically analyzed" [Ant+14]. As a case study an example based on "concerns a regulatory organization that assesses and monitors the structural safety of large engineering infrastructures, such as hydroelectric power

plants, dams, and bridges" [Ant+14] is provided. The proposal "was evaluated with a case study that uses ArchiMate as the upper ontology and a number of domain-specific ontologies that extend the core description language" [Ant+14]. The authors claim that "in this way, the domain-specific ontologies increase the expressiveness of the upper ontology with domain-specific aspects" [Ant+14].

Another very important work that connects EA and graph-based analysis comes from Naranjo, Sánchez, and Villalobos and their prototype tool PRIMROSe [NSV15]. The authors highlight the problems of complexity and size when it comes to static analysis of models. They discuss how *"current approaches focus on partial views and queries over this model, leading to partial assessments of the architecture"* [NSV15] and propose *"a different approach to EA analysis, which consists on the incremental assessment of the architecture based on the interaction of the user with visualizations of the whole model"* [NSV15]. So the main idea is to move from partial view-based assessment to whole model assessment. Their approach is based on graphs which are *"a homeomorphism on the enterprise models"* [NSV15]. In PRIMROSe, Naranjo et al. are focusing on: Incremental Analysis, Reusable and extensible functions, Non-destructive Analysis, Independence from the Visualization toolkit, and Customizable Visualizations. The PRIMROSe architecture can be seen in Fig. 3.3. It consists of four components: Graph Manager (used to translate EA model into an Expanded Graph), Project Manager (administration of the different projects of a user), Pipeline Engine (management and application of different stages with respective sequential operations), and User Interface (editor for Pipeline Descriptors and workbench for visual analysis).

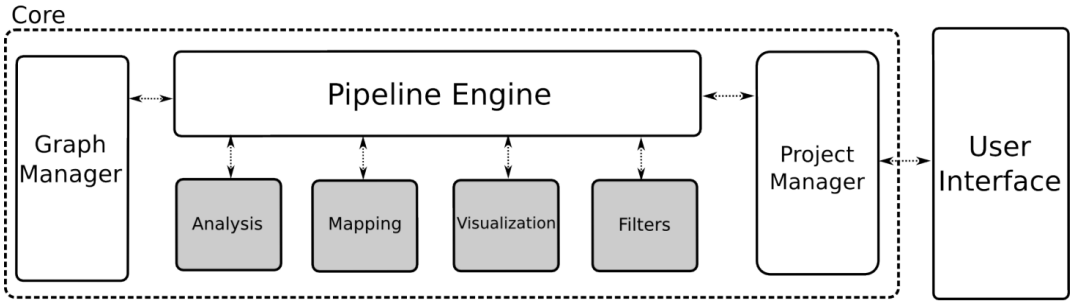


Figure 3.3: PRIMROSe Architecture. [NSV15]

An example of PRIMROSe model visualisation can be seen in Fig. 3.4. Authors also define term *Analysis Functions "which receives as input the Analysis graph"*, and *"its output is the modified graph, with additional selectors, and even new attributes"* [NSV15]. As an example they implement two analysis functions: Degree Calculator and Impact Analysis. The result can be seen in Fig. 3.5.

More recent non-academic work seems to really address the intersection of EA and graph structures. Lazarevic speaks about on *Using a Graph Database to Explore Your ArchiMate Model* in her blog pages [Laz19; Laz21]. Lazarevic addresses "the challenge in getting started can be from moving from existing modelling tools and artefact repositories to be able to start using the power of graph databases" [Laz19]. Lazarevic also talks about

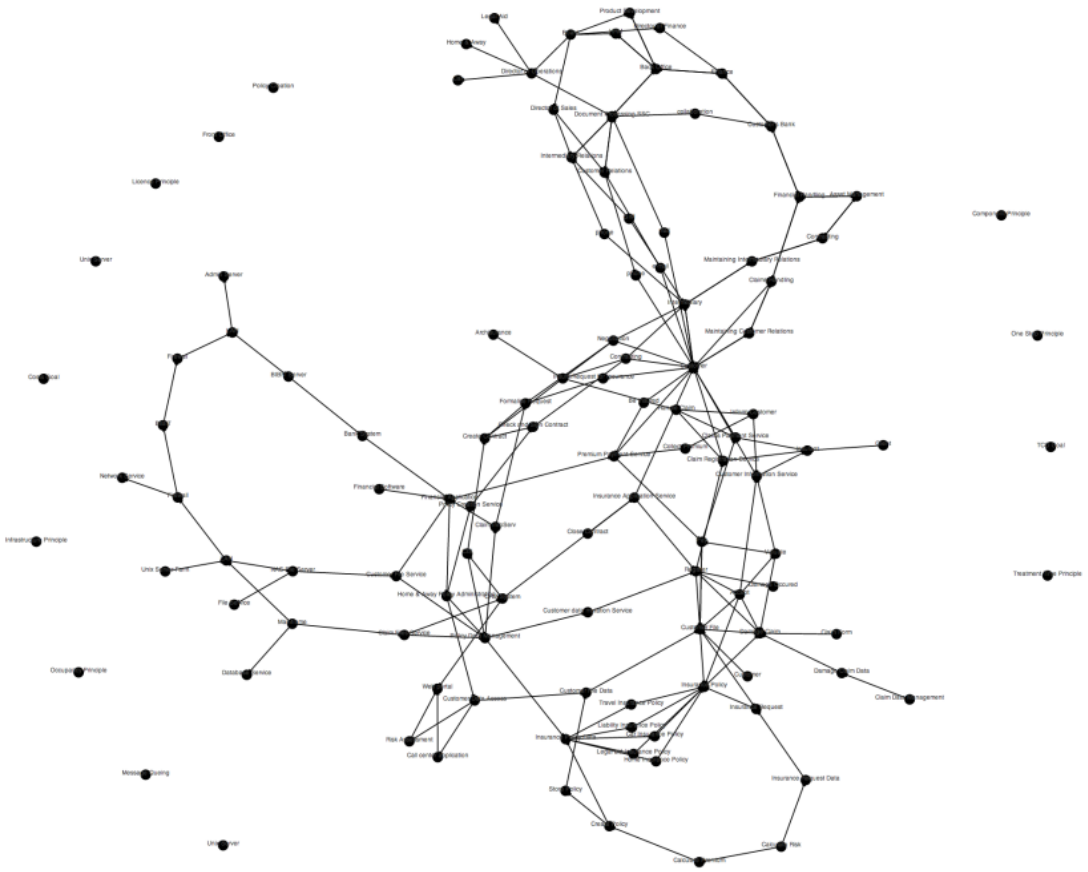
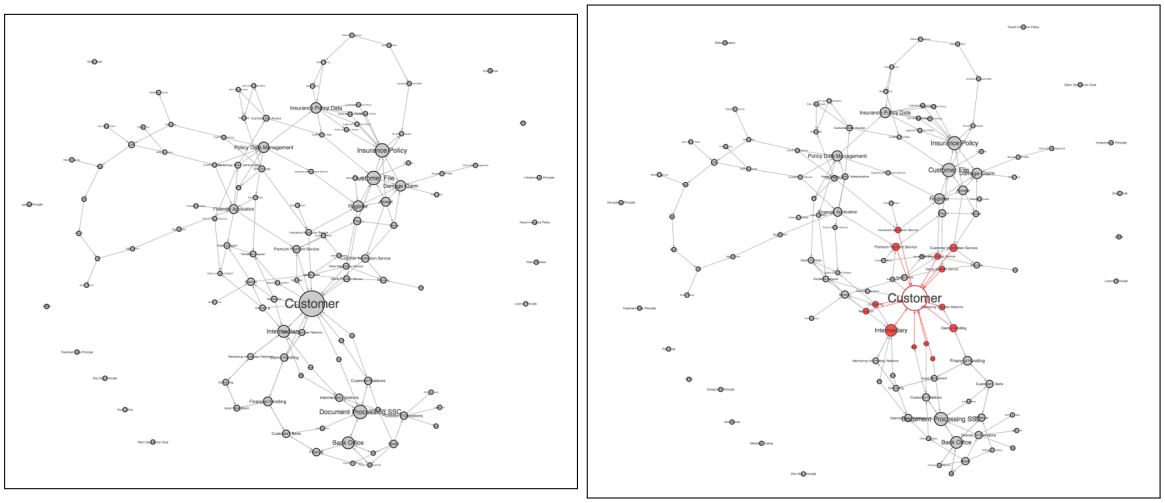


Figure 3.4: PRIMROSe ArchiSurance Model Graph. [NSV15]

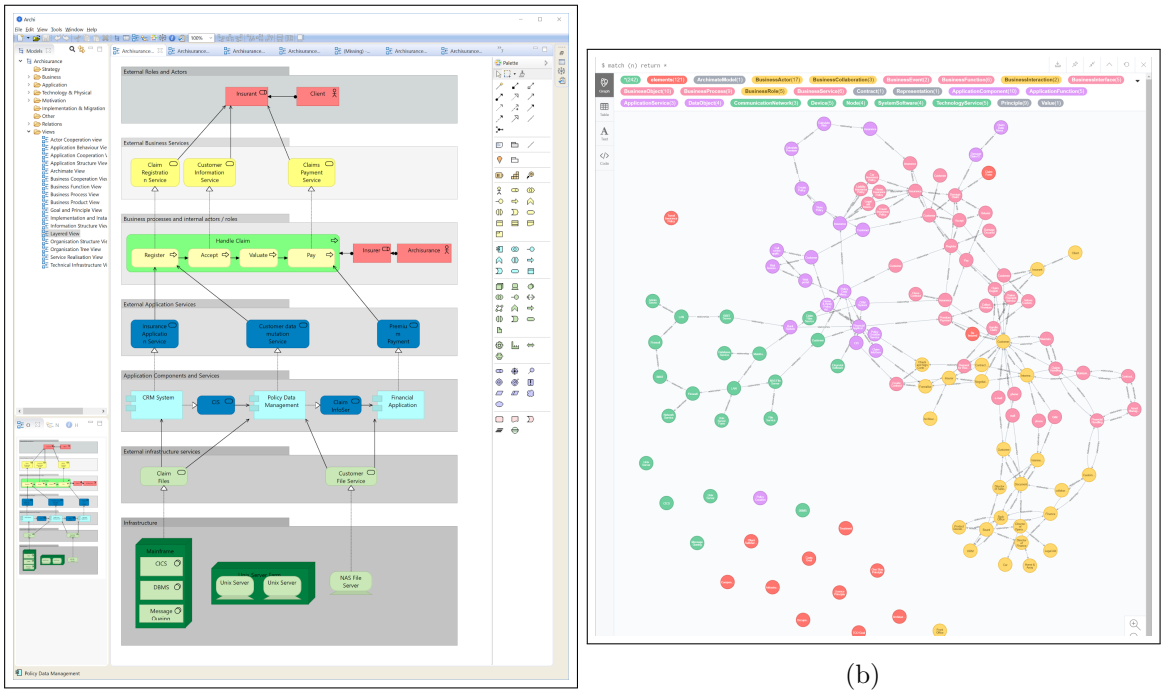
importing "an existing ArchiMate diagram into Neo4j using either the Archi database plugin, or Neo4j Cypher queries" [Laz21] and then "query the diagram, both for basic items such as lone elements and connectivity, to using some graph algorithms to look at element relationship strengths" [Laz21]. In order to import the data in Neo4j she mentions two ways. First one via the Archi database plugin and second via the Neo4j CSV importer. Archi database plugin [Jou+21] provides the possibility to export Archi models to PostGreSQL, MySQL, MS SQL Server, Oracle or SQLite and Neo4j [Jou+21]. An example can be seen in Fig. 3.6.

She then provides examples of questions an enterprise architect could ask and provides a Cypher query to answer this question. One such example is *"Q4: What elements are impacted if technology service "Claim Files Service" stops working?"* [Laz21] which is answered by Cypher query matching pattern. Moreover she even uses some centrality algorithm measures in order to answer some questions. One example for this is *"Q5: What are the most connected elements in my estate?"* [Laz21] where she uses PageRank algorithm to answer this.



(a) (b)

Figure 3.5: Incremental appliance of Analysis Functions Degree calculator (a), Impact Analysis (b). [NSV15]



(a) (b)

Figure 3.6: Neo4j Archi database plugin transformation of initial model (a), result in Neo4j (b).

Study	Technology	Tool support	Modelling language	Approach
[GKC06]	WEB J2EE	IDEA	-	Graph visualisation
[Aie06]	Desktop Java	EA builder	specific *EA Builder* meta-model	Clustering algorithms in networks
[IJ06]	-	-	ArchiMate	Quantitative Analysis
[LH12]	-	-	BPMN	Network analysis
[Ram+14]	Desktop	ArchiAnalysis	ArchiMate	Quantitative analysis
[SFM16]	-	-	TOGAF	Survey network analysis
[Hol+09]	-	GeNIe-tool	ArchiMate	Bayesian Belief Network
[ÖLR12]	Desktop	TEAMATe	ArchiMate, TEAMATe metamodel	Metrics, OCL
[SS15]	-	-	ArchiMate	Lightweight graph metrics
[BJS13]	Desktop Java	EAAT	customized ArchiMate	Property analysis, fault tree formalism
[ACB14]	-	-	ArchiMate	Description Logics
[Ant+14]	-	-	ArchiMate	OWL-DL
[NSV15]	Desktop	PRIMROSe	ArchiMate	Graph based analysis
[Laz19; Laz21]	Desktop	Archi database plugin	ArchiMate	Graph based analysis, Neo4j
[Hew19]	Desktop	Archi database plugin	ArchiMate	Graph based analysis, Neo4j, Gephi

Table 3.1: Summary of studies with a relation between EA analysis and graph structures.

Heward reflects on [Laz19] in his blog page [Hew19] and introduces more ideas. He wanted to "be able to get data out of the excellent Archi tool into a place where I can perform meaningful queries on it" [Hew19]. He then explains how he was thinking to achieve it. He talks about the idea to use the Jasper reporting tool, but that turned out to be too complicated. Moreover, he talks about the possibility to export Archi model to Open Exchange File and use some XLST transformation to transform the data to GraphML and import it to Neo4j. However he had issues to load GraphML file to Neo4j. Then he tried to use Gephi, but this also did not enable much possibilities to querying for him. He also talks about importing it to MySQL. He also compares the Neo4j to JArchi and MySQL database and he claims neo4j "is just more dynamic and visual" [Hew19] and it *"really works well with highly complex data structures (which a*

mature Architecture repository based in ArchiMate becomes)" [Hew19]. He emphasizes that Neo4j provides "queries and dependency queries that are not naturally possible with scripts and SQLs" [Hew19].

In the end of this section we provide a summary of different studies mentioned before. Summary can be seen in Tab.3.1. We set this summary based on following criteria:

Technology: which technology is used in tool

Tool support: if exists, name of the tool developed to support analysis process

Modeling language: which modeling language is used to represent the architecture

Approach: high level representation of the used analysis approach

3.2 State of the art EA Tools

Apart from different analysis approaches from various research papers, standardized industry tools compete to be selected by large enterprises in real-world situations. Many tools enable EAM in organizations. Choosing a state-of-the-art tool at a given time is no easy task. Since having information on what tool is among the best ones is important to different stakeholders, many consulting agencies exist that research to give clear and precise recommendations to their clients. One such company is Gartner [Gara]. Gartner is one of the world's most popular review agencies, and many companies compete to get a better ranking there. Gartner offers what they call *"Magic Quadrant"* that consists of two dimensions:

- Ability to execute
- Completeness of vision

These two dimensions then enable 4 quadrants:

Challengers high ability to execute, low completeness of vision

Niche players high ability to execute, low completeness of vision

Visionaries low ability to execute, high completeness of vision

Leaders high ability to execute, high completeness of vision

Companies placed in the quadrant by Gartner use this as one of the most vital marketing points. One such example is ServiceNow [Ser] with *ITSM Magic Quadrant*. Gartner has also done the same for Enterprise Architecture Tools for many years now. The latest version *Magic Quadrant for Enterprise Architecture Tools 2021* [Garb] presents what

Figure 3.7: Magic Quadrant for Enterprise Architecture Tools [Garb].

they claim the state of the art tools for Enterprise Architecture. Results for Enterprise Architecture tools can be seen in Figure 3.7

The *Leaders* (state of the art) based on Gartners in Enterprise Architecture tools are:

BOC Group Tool name: ADOIT. Slogan: "*Leverage Enterprise Architecture to Outpace Your Competition.*". Key features: ArchiMate, Lean Design & More, Effective Teamwork, Agile Goal Management, Powerful Analysis ("*Leverage endless graphical analysis possibilities. Turn data into actionable insights. And facilitate fact-based decision making across the enterprise.*"), Next-Level Openness, Flexible Set-Up &

Licenses, Great Configurability, Easy Migration [Gro].

LeanIX Tool name: LeanIX EAM. Slogan "*Manage the transformation and risk of your IT landscape*". Key features: "increase speed (access to relevant data immediately, save time for manual data collection, accelerate decisions with insightful data), save costs (reduce redundant or unused technology, identify cost drivers quickly, free up financial resources for innovation), reduce risks (identify technical debts easily, address technology risks proactively, stay compliant with regulations)" [Lea].

MEGA International Tool name: MEGA, Slogan: "*Leverage next-gen enterprise architecture solution to accelerate business transformation*". Key features: IT Business Management, IT Architecture, IT Portfolio Management [Int].

BiZZdesign Tool name: BiZZdesign. Slogan; "*Integrating IT insights for a holistic approach to executing business strategy and business change*". Key features: "Native ArchiMate 3 support for consistent modeling, Collaboration across departments and continents, Support decision-making with customizable views and dashboards, Define and manage the impact of change" [BiZc].

Software AG Tool name: Alfabet. Slogan: "*Change enterprise architecture fast with Alfabet*". Key features: Validate change initiatives, Coordinate your changes, Inform change decisions [AG].

Avolution Tool name: ABACUS. Slogan: "*ABACUS enables organizations to analyze business scenarios, model people, processes and technology, build roadmaps and align IT and business strategies*". Key features: "Cloud-based collaboration, supporting remote working and digital strategy, Building enterprise architecture roadmaps to navigate IT and business change, Centralized data in a single repository with our integrations and API" [Avoa].

Capsifi Tool name: Capsifi. Slogan: "*Align organizational teams around a common understanding of how business functions are supported by your technology portfolio and manage the lifecycle of your digital assets with Capsifi's enterprise architecture operating model.*". Key features: "Transformation planning, Strategic roadmaps, Business motivation model, Value trees, Capability maps, Heatmaps, Technology portfolio, Customer segments and personas, Organisation structure, Business requirements, API management, etc."[Cap].

Ardoq Tool name: Ardoq. Slogan: "*Navigate Your Change Initiative with Confidence*". Key features: "Automation, Engagement, Flexibility, Surveys, Presentations, Broadcasts, Graph Powered Reporting and Dashboards, Intelligent Graph Powered Fields, Graph Powered Filters, Visualize, Compare and Share at Speed" [Arda]. This tool provides: *graph powered reporting and dashboards*, *intelligent graph powered fields*, *graph powered filters* which will be examined further in the evaluation chapter.

Based on the Gartner Magic Quadrant for Enterprise Architecture tools, we will compare some leaders and open-source tools like Archi. The following sections will briefly explain the analysis possibilities of some open-source and Gartner leaders tools.

3.2.1 Archi

We will start with Archi. Archi tool is an open-source tool and is running on the local machine. On the left side, a user can see the *models* view where all the elements and views are located. Users can search by element name or view name or relation name. When some element is selected in the view, a user can see in which views the element is used and which relations it has in a list manner. Next, in the *Visualiser* user can check the relations of the element in a graph-based manner.

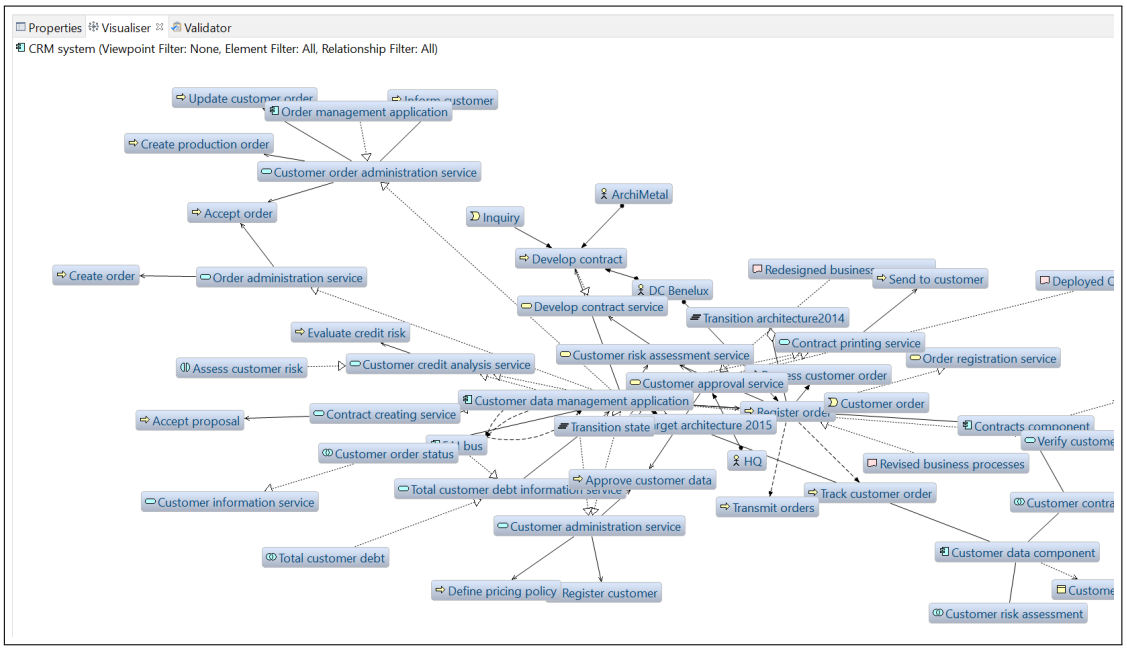


Figure 3.8: Archi Visualiser view with the possibility to set relation depth.

What is important to note is that it is possible to select the depth of the visualization to up to 6 (this means the user can see all the relations of the selected elements up to the 6th element following the relations). On the same visualizer, a user can also see the relations of the neighbors of the selected element.

Regarding the filtering of this visualization, it is possible to select which element type (user can select to use all or only one element type) is used, which relation type (same applies as for elements), and also which relation direction. Additionally, for this graph-based visualization, it is possible to select specific *Viewpoint filter* which are sets of specific element types and relations. One example is *Application Usage* viewpoint filter

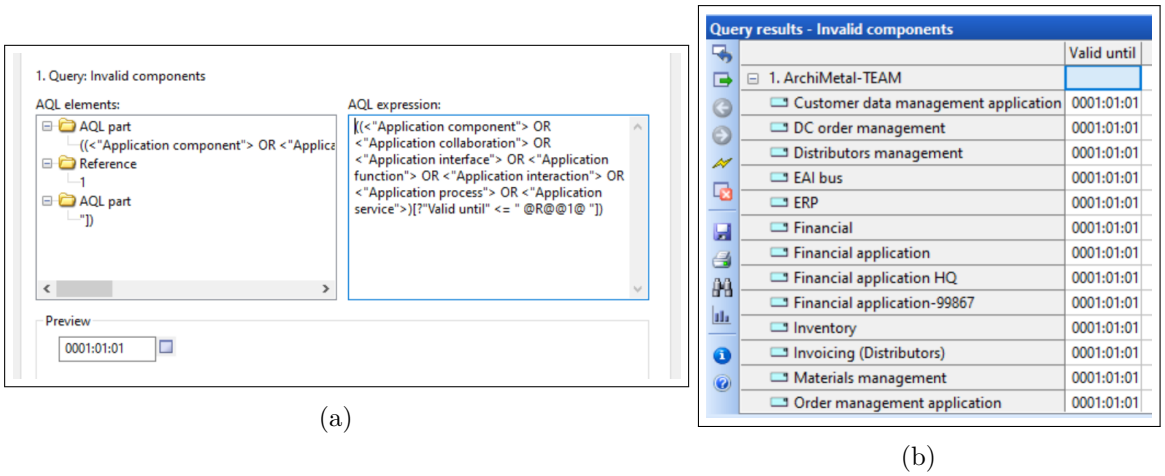
that will show relations to elements like application service, business process, application collaboration.

The tool offers the export of created visualizations as images. Additionally, the tool offers *Validator* tab which will show errors, warnings, and advice. An example of an error is the *illegal* relation, and an example of warning is that the element is not used in a view or element is a possible duplicate. An example of advice would be that the view is empty or the element is nested visually in another element, but there is no nesting relation between them.

Apart from standard functionality there are plugins and one such plugin is discussed in Sec. 3.1. This plugin [Jou+21] can export the Archi model to database engines like PostGreSQL, MySQL, MS SQL Server, Oracle or SQLite and Neo4j.

3.2.2 ADOxx and TEAM

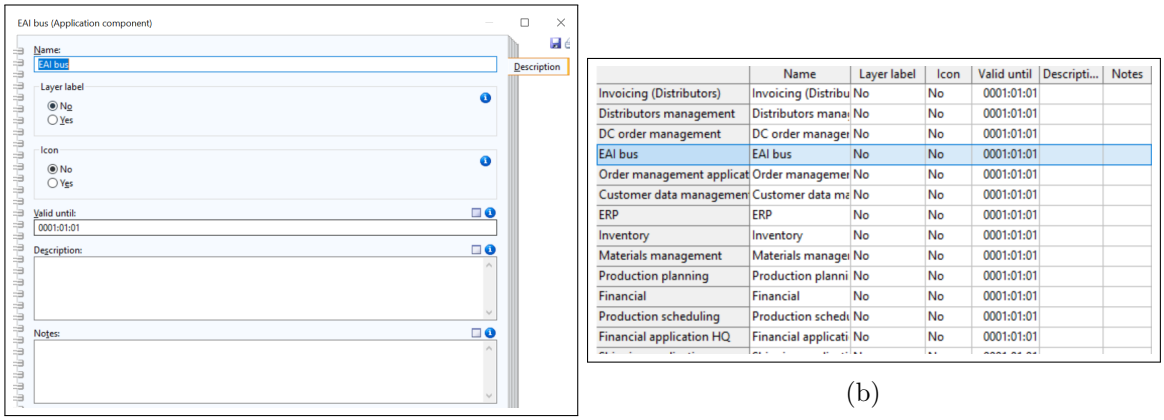
Another open-source tool we will evaluate is ADOxx. The same as Archi, it runs locally on a machine. The difference to Archi is that ADOxx uses a local SQL database to store all information. This includes information about users, libraries, and models. ADOxx provides the possibility of creating a custom meta-model, and using it to create model instances from it. The platform itself consists of two parts: ADOxx Development Toolkit and ADOxx Modeling Toolkit. Users can manage users, libraries, and models in the development toolkit. On the other hand, the modeling toolkit provides the possibility to work with the models, analyze models, and execute simulations and evaluations. In the context of analysis ADOxx provides a separate button called *Analysis* which provides the following features: *Queries/Reports*, *Predefined Queries*, and *Analytical Evaluation*. Queries are defined in the development toolkit and an example can be seen in Fig. 3.9.



(a) (b)

Figure 3.9: ADOxx TEAM library example query definition development toolkit (a), result in modeling toolkit (b).

Furthermore, in the same way, as for executing *analysis queries*, ADOxx provides the possibility to create *evaluation queries*. With respect to visualizations, ADOxx does not offer different model layouts. It does provide a *Notebook* and *Table* view for entity. This can be seen in Fig. 3.10.



(a) (b)

Figure 3.10: ADOxx TEAM library example notebook view (a), table view (b).

On the other hand, users can create different icons for each entity and relationship, resulting in the possibility of creating the ArchiMate model. This was enabled by the TEAM tool [Bor+18]. The tool was a joint work of Bork et al. where they have created "a prototype TOGAF-based Enterprise Architecture Management (TEAM) modeling tool that implements the Archimate 3.0.1. standard" [Bor+18]. The TEAM modeling tool is realized with the ADOxx metamodeling platform, and it supports all ArchiMate 3.0.1 layers aligned with the TOGAF framework. Combined, the tool "enables basic ArchiMate modeling and TOGAF support as well as acting as a facilitator for EAM" [Bor+18].

The tool offers *"the modeling palette, listing the available ArchiMate language concepts of the currently opened model on the left side"*, *"intuitive context menu that features the model queries - e.g., for the lifecycle management"*, and *"the additional functionality - e.g., for the business continuity management"* [Bor+18]. The screenshot from the tool can be seen in Fig. 3.11.

3.2.3 ADOIT

This tool runs directly in the cloud and is accessible via the browser and is accessible via user credentials. Straight from this, it can be seen that sharing is easy since the tool is running in the cloud and has user management. On the left side, the tool offers three tabs: objects, models, quick access. Each tab contains a list, and it is similar to Archi. On top of these tabs, there is a field filter, where the user can type in some text, and objects/models will be shown. Users can also download the models and objects in the ArchiMate exchange file.

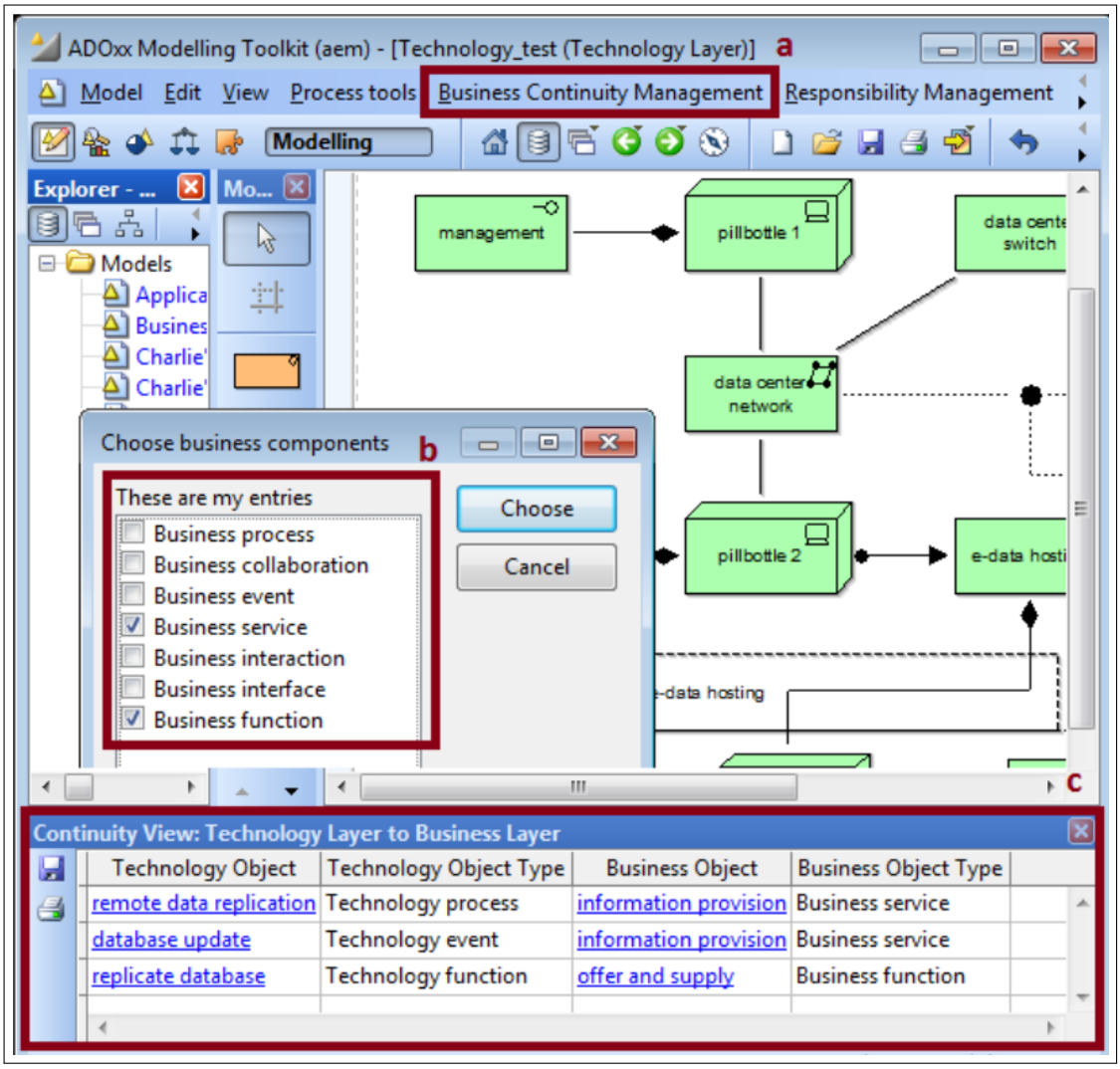


Figure 3.11: "Executing model queries in the TEAM tool." [Bor+18]

Users can also download the report in PDF format containing the model's image and all elements with details and relationships. Furthermore, the interface has *Analyse* button in the top navigation bar. This offers reports in the following representations: cluster map, Gantt, matrix, and portfolio. Additionally, there are predefined reports grouped by the ArchiMate layer. For example, users can create a Gantt report with all the application components and system software to see when some software will expire. Another example is to create a matrix to see which business object uses which application component.

On the other hand, the tool offers search with the following criteria: topic(motivation, strategy, business), model (by name), object (by name), by property (name or description). The tool also offers to validate button, which can be executed on all models. This will

provide information about methodological warnings such as the mandatory field is not populated or relation does not comply with best practice. There is a button *Insights* for each object where the user can see the classification of the element in RACI, relations to other elements in graph-based view, data usage in applications, and application component details with the business and IT fit.

3.2.4 ABACUS

Another tool is Avolution ABACUS. It provides many standard frameworks like ArchiMate, BPMN, and many more (100+ frameworks). Each meta-model can be adjusted. Although ABACUS is a desktop tool, it provides cloud-based collaboration. On the analysis side, it provides scenarios similar to other tools in order to test future state architectures.

It also provides automated calculations to show costs, risks, technical and operational metrics, and dependencies. In order to visualize things, ABACUS offers to create dashboards with charts and metrics specific to different stakeholders. ABACUS has used graph database technology for over 15 years [Avob], where they claim that traditional relational databases are a source of frustration because they cannot fulfill increasing user demands and analysis techniques.

3.2.5 HoriZZon

BiZZDesign offers a tool called HoriZZon, which is an enterprise architecture platform. The platform has a modeling environment called *Enterprise Studio*. The platform is a SaaS product, but also hybrid (cloud and desktop), and also on-premise versions are available. It provides integrations to Jira, SAP, Azure, AWS, and others regarding data integration.

Modeling and roadmapping as core feature, offers different modeling frameworks like ArchiMate 3.1, BPMN 2.0, UML/ERD, DMN. Out-of-the-box platform offers different *user stories*. These user stories are *Strategy and Motivation*, *Understand*, *Connect Business Technology*, *Design Roadmap Change*, *Assess Risk*, *Execute Change*. Each user story is a set of different views for management purposes.

For example, *Strategy and Motivation* contains views like *Transformation Dashboard*, *Business Motivation Model*, *SWOT Analysis*, and *Balanced Scoreboard*. By default, the views are visualized using Elasticsearch Kibana. Other BI tools are also supported. When an entity is selected on the right-side, a user can see properties, documentation, metrics, relations, and views.

An example of metrics for an ArchiMate capability is *Total OpEx(Current)*. These metrics can be then incorporated into different charts, and all these charts can be exported. In the *Enterprise Studio*, it is possible to select different elements and create a view that contains relations to predefined types of elements.

Figure 3.12: Avolution ABACUS graph database usage vision of benefits [Avob]

3.2.6 Ardoq

Ardoq and ABACUS are probably the closest to what this research is about. Ardoq is *data-driven enterprise architecture platform* which is powered by graph-based analytics. The core thing in this online platform is the graph that represents the enterprise architecture modeled by different stakeholders. This graph consists of *components* and *references*, which represent nodes and relations, respectively. These components and references are contained in workspaces that act as repositories.

The platform offers out-of-the-box meta-models (such as ArchiMate 3.0) and allows the

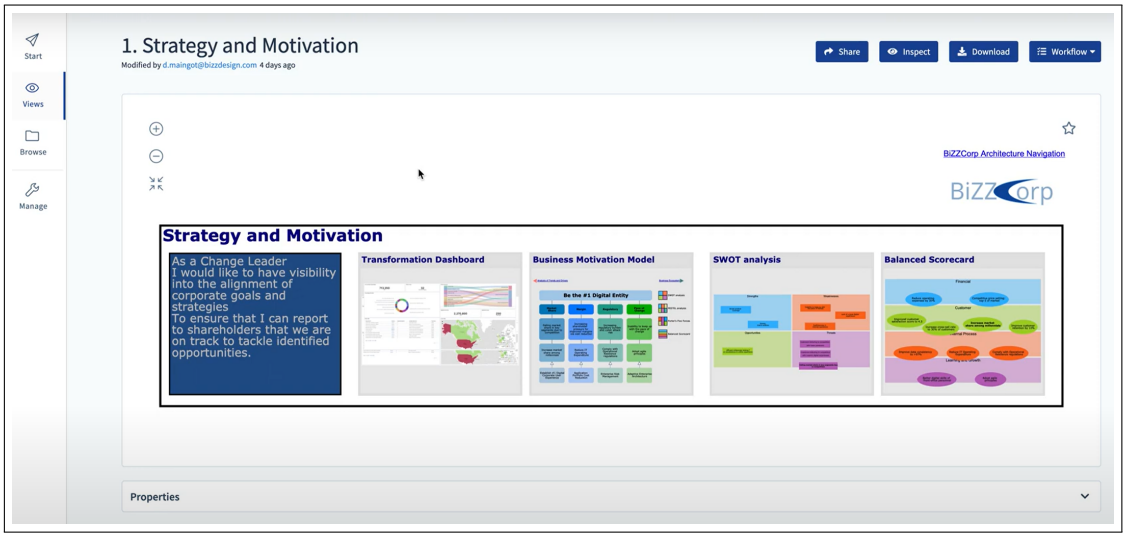


Figure 3.13: Strategy and Motivation user story views in HoriZZon [BiZa]

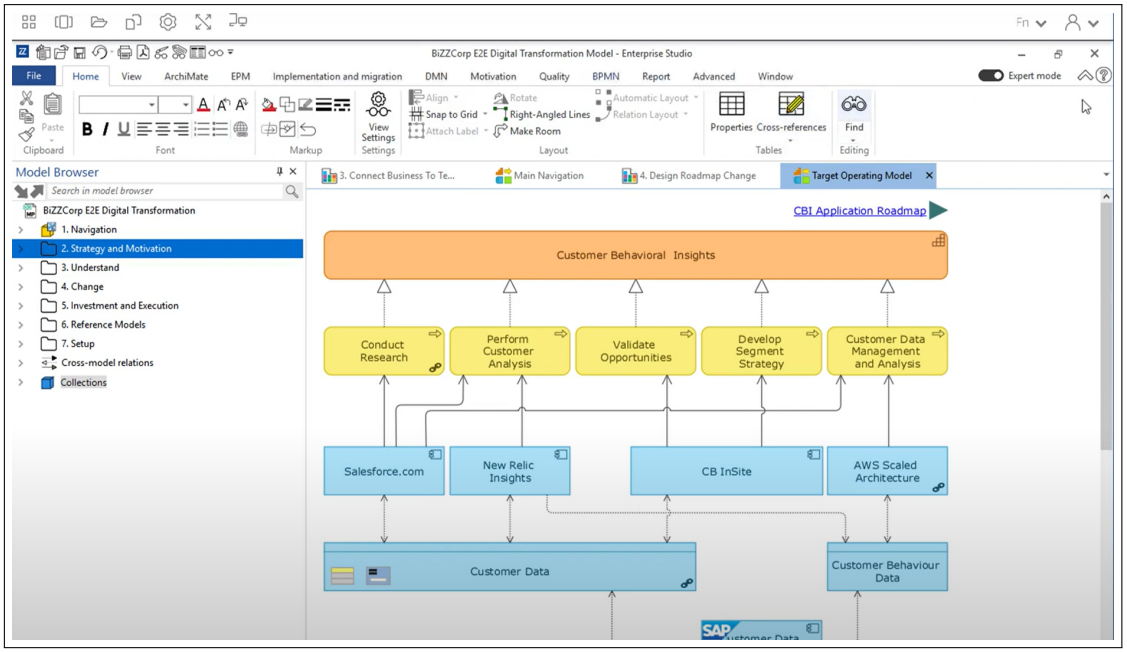


Figure 3.14: Enterprise Studio in HoriZZon platform [BiZb]

creation of a new meta-model that can be used later. Furthermore, the tool offers creation of *dashboards* and also *presentations*. Compared to other tools we mentioned, these presentations are unique about this platform. These presentations offer the possibility to create different views with the reports based on the graph behind them. These reports consist, similar to ADOIT, gent diagrams, different matrices, and others.

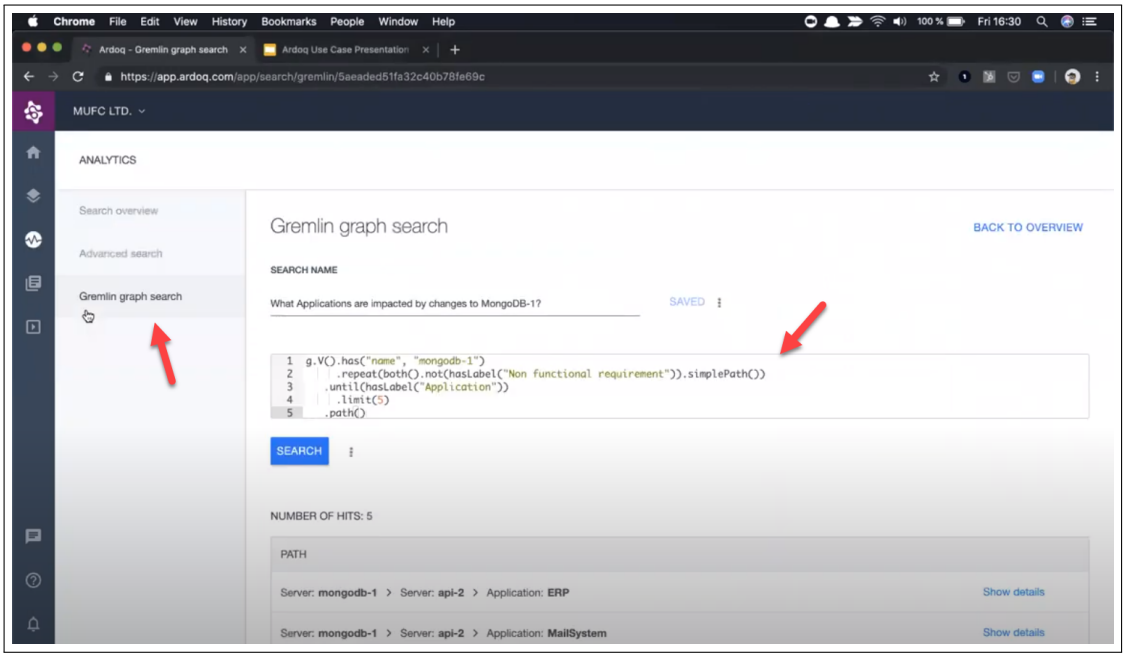


Figure 3.15: Possibility to define Gremlin search queries in Ardoq [Ardc]

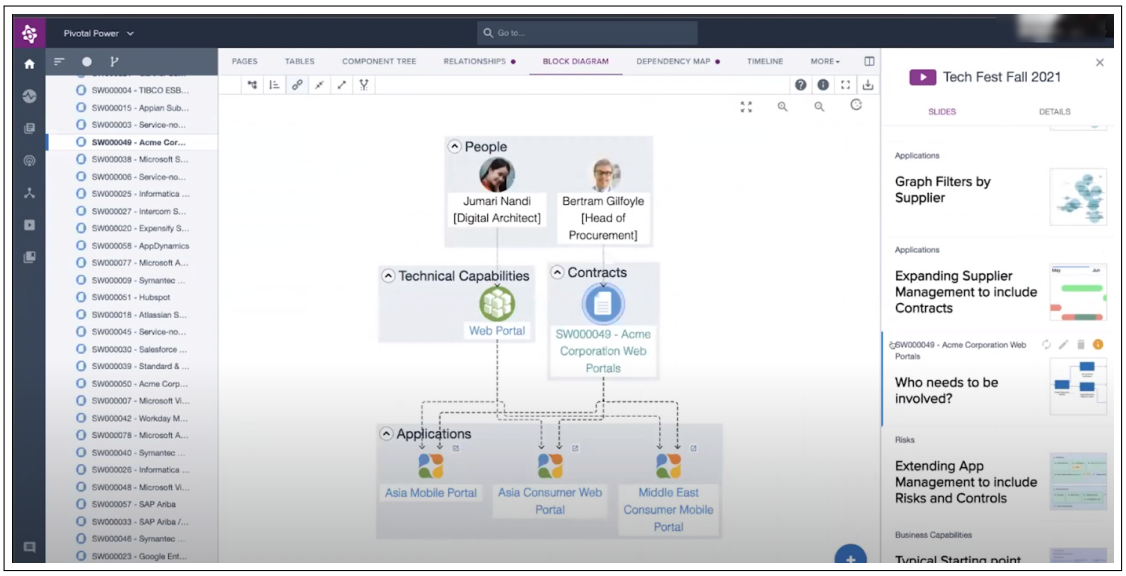


Figure 3.16: Block diagram with specific components and references grouped by component type in Ardoq [Ardb]

Users can also create a view based on selected components and relations that can be further grouped in such a view based on different conditions. These conditions can be component type based, property condition-based. The tool has many integrations to

different sources of data like Excel, Jira, ServiceNow, and others. Users can easily share created dashboards and presentations with other users. It is also possible to directly run the presentation in the browser and show it to people. It also has some other features like surveys and architecture scenarios that can be easily visualized to as-is and to-be architecture models.

As already mentioned, the platform stores all the entities and relations in a graph database. This leads to another possibility to provide execution of the queries directly. Luckily this is also enabled. The platform provides a possibility to specify queries written in Gremlin and execute them on the database. There is a set of predefined queries, but users can also create/save/share queries with other users. Considering this tool, it acts most similar to the platform developed in this thesis as a prototype.

3.3 Summary

If we compare Fig. 3.1, Fig. 3.2, Tab. 3.1 and Sec. 3.2 we can conclude the following. There are many studies with different initiatives that utilize network analysis for analyzing EAs [SFM16]. Additionally, multiple statements and recommendations are highlighting the need for EA analysis tool development [SFM16]. Tab. 3.1 shows that there is not much diversity in modeling language (mostly ArchiMate) and is mostly a desktop-based tool. On the other hand, state-of-the-art tools are all cloud-based, offering dozens of different modeling languages and have uniform analysis methods. This leads us to a conclusion that there is a high need for a generic tool (platform) that will support the implementation of different initiatives discussed in [SFM16]. This is the main motivation for this thesis. To build up such a tool, we first have to define a framework that will serve as a starting point in the graph-based analysis of EA models. We state here *EA* models, but we can also extend this to a more general topic, i.e., conceptual models. We introduce the framework that explains how the general graph-based analysis of conceptual models could be. The central part is the connection between conceptual models themselves and network structures. This central part is the transformation of the conceptual model to a suitable network analysis-ready structure, which we introduce in the following chapter.

CHAPTER 4

Transforming Conceptual Models to Graphs

In this chapter, we will discuss the design of a general framework for analyzing conceptual models in a graph-based manner. We will also define a transformation from different meta-models to graph structure meta-models.

4.1 General Transformation Concept

This section introduces the framework to analyze the EA in a graph-based manner. The framework can be seen in Fig. 4.1. Opposite to other works, this framework is designed to be generic and easy to extend. It consists of three parts. It starts with popular conceptual models developed in Ecore or ADOxx meta-modeling platforms. This way, the framework enables the transformation of all models created with these meta-models to a graph structure model. Next, the central part is the transformation of the model to a GraphML model itself. This includes high-level transformation and also domain-specific transformation if needed. This all together enables the use of third-party tools used to analyze graphs. These tools have to support standardized graph structure GraphML. This way, we design a framework that has models produced in standardized meta-modeling platforms on one side, and the other side is an analysis of the graph itself. Having this approach will enable a unified graph-based analysis approach of different models.

As seen in the framework overview Fig. 4.1, it was designed to start from meta^2-level to provide general transformation rules for higher-level (meta^2) and then define custom rules, if necessary, for lower meta-level. It can be discussed why it did not start from the meta^3 level but this can be left for further research. Additionally, we focus on the most popular meta-modeling platforms to show the framework's generality. After the model is transformed (either from meta^2-level or meta-level), the result is a graph-based structure

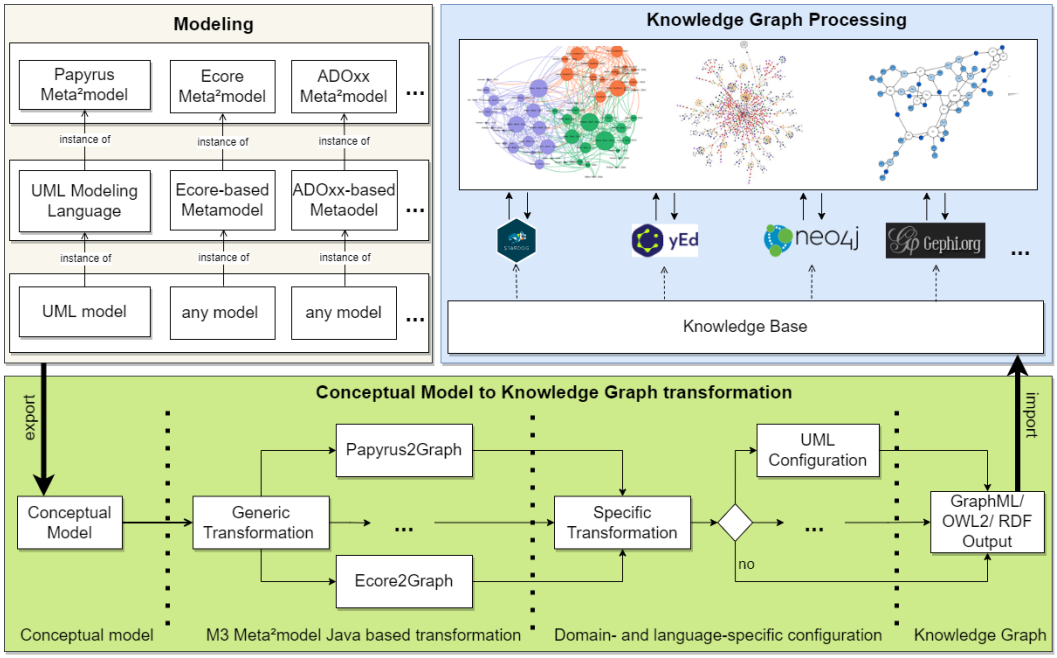


Figure 4.1: A generic framework for transforming conceptual models into graphs. [SB21a]

file in standardized GraphML format. This format is supported by popular and powerful tools such as Neo4j, Gephi, and yED. Some examples were displayed in the modeling section in Fig. 4.1, which means we will discuss this further. For these examples, we will provide transformation rules from Ecore, Archi, ADOxx, and PapyrusUML models into GraphML. We take these four types of models to show the applicability of our generic framework.

4.2 Ecore to GraphML Transformation

The first transformation is a transformation from the Ecore model to the GraphML model. Meta-model to meta-model mappings can be seen in Fig. 4.2. It can be clearly seen which Ecore concepts are mapped to which GraphML concepts. First *EPackage* is being mapped to *GraphML*, and then all others are mapped to *Graph*. Every *EClass* corresponds to *Node*, and each *EReference* corresponds to an *Edge*. For *source* value of an edge, we take the initial EClass, and for the *target*, we take the corresponding outgoing *eReferenceType*. Additional information about the EClass and EReference by means of *EAttribute*s is stored in the *Data* concept. Mapping of Ecore to GraphML concepts in summarized version can be seen in Tab. 4.1.

Regarding the implementation of mapping shown in Tab. 4.1 there can be different ways. Here we discuss one possibility where we will utilize existing methods of Java implementation of Ecore metamodel [ecl21]. Algorithm 4.1 presents the pseudo-code for a given Ecore2GraphML transformation. The transformation uses any specific conceptual

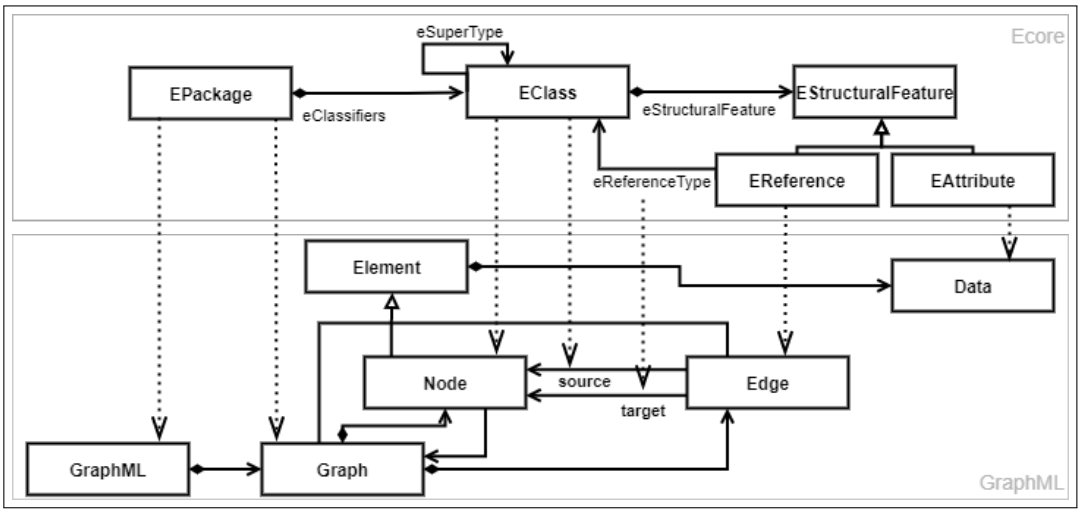


Figure 4.2: Generic transformation from Ecore to GraphML. [SB21b]

Ecore Concept	**GraphML Concept**	**Rationale**
EPackage	GraphML	General graph placeholder.
Epackage	Graph	Different packages apply to different graphs.
EClass (EC)	Node	Ecore class represents graph node.
EReference(ER)	Edge	Ecore reference represent graph edge. $EC \rightarrow Edge.Source$ $ER.eReferenceType \rightarrow Edge.Target$
EC.EAttribute	Node.Data	Applies for all nodes.
ER.EAttribute	Edge.Data	Applies for all edges.

Table 4.1: Ecore2GraphML transformation rules.

model that is an instance of the Ecore metamodel as input and outputs a corresponding graph represented in GraphML. As said, algorithm takes instance of model developed by Ecore metamodel and outputs GraphML instance. In order to do so, there are three important parts. The first part is to iterate through all packages of the input model (line 1-20. 4.1). We map this to GraphML graph concept. The second port is to iterate through all EObjects and transform them into nodes (line 3-9. 4.1). It is also important to take all attributes of EObject and add them to the already created node (line 5-7. 4.1). Third and last part is to transform the relationships between EObjects into edges. We have to iterate through all EObjects again (line 10-20. 4.1) and iterate all its relationships (line 11-19. 4.1). We did not do this in the second part because to create an edge in GraphML, we have to have all nodes already created. In order to create an edge (line

12.), we have to set source and target values, for which we have to find an already created node (method *findNode* lines 13. and 14.). After this is done, same as in step two, we also have to take special attributes of Ecore reference and transform them into GraphML edge attributes (line 15-17. 4.1). If this is all done, we should have transformed all information from the input model into the output model.

Algorithm 4.1: Ecore2GraphML transformation.

```
   Input: Ecore model instance.
   Output: GraphML model instance.
 1 for EObject package : input.eAllContents().getPackages() do
 2 |   Graph g ← transformPackage(package)
 3 |   for EObject eo : package.eAllContents() do
 4 |   |   Node n ← transformNode(eo)
 5 |   |   for EAttribute a : eo.getEAllAttributes() do
 6 |   |   |   n.addAttribute(transformAttribute(a))
 7 |   |   end
 8 |   |   g.add(n)
 9 |   end
10 |   for EObject eo : package.eAllContents() do
11 |   |   for EReference ref : eo.getEAllReferences() do
12 |   |   |   Edge ed ← transformEdge(ref)
13 |   |   |   ed.source ← findNode(eo)
14 |   |   |   ed.target ← findNode(eo.get(ref))
15 |   |   |   for EAttribute a : ref.getEAllAttributes() do
16 |   |   |   |   ed.addAttribute(transformAttribute(a))
17 |   |   |   end
18 |   |   |   g.add(edge)
19 |   |   end
20 |   end
21 |   output.add(g)
22 end
23 return output
```

4.3 ADOxx to GraphML Transformation

In this section transformation of a model developed in ADOxx is discussed. The transformation rules for this transformation are given in the table 4.2. As seen in Fig. 2.8 in the middle of the ADOxx metamodel, there is a model type that is the base of the model which has attributes and contains objects and relationships together with their attributes. Therefore the mapping goes from ADOxx *Modeltype* to GraphML *Graph*. For each model we transform *Class* to *Node* and Relation *Class* to *Edge*. Additionally we also transform class and relation class *Attributes* to *Data*. It is important to note here

the particular case of classes with type *Grouping* which can hold other class instances; therefore, we map this to a Graph.

ADOxx Concept	**GraphML Concept**	**Note**
-	GraphML	General graph placeholder.
Modeltype	Graph	Different model types apply to different graphs.
Class "Grouping"	Graph	Classes of type "Grouping" represent subgraphs.
Class	Node	Class represents graph node.
Relation class (RC)	Edge	Relation class represent graph edge. *RC.is From* → *Edge.Source* *RC.is To* → *Edge.Target*
Class.Attribute	Node.Data	Applies for all nodes.
RC.Attribute	Edge.Data	Applies for all edges.

Table 4.2: ADOxx2GraphML transformation rules.

ADOxx uses a standardized and unified Extensible Markup Language (XML) structure for storing and transferring the models from one environment to another. This is not the case with Ecore, which is always different and dependant on an actual conceptual model when exported to XMI. Therefore the transformation works by transforming the ADOxx XML file to a GraphML. The pseudo-code is given in algorithm 4.2.

Here we utilize the XML structure of ADOxx file, and therefore we can iterate through the nodes of the XML document and transform it how we want. The algorithm has two main parts here, and each part has specific rules to handle the specific case of grouping we mentioned before. The first part consists of iterating through all nodes of the XML document with the name "INSTANCE" (line 2-12 4.2). For each instance, we check whether it is grouping or not. In case of grouping we create new graph (subgraph) and add it to the parent graph (line 3-5 4.2). In other cases, we create a node and map its properties (line 7-11 4.2).

After the first part is completed, i.e., creating the nodes and subgraphs, we can now iterate through connectors and build up edges. Again here, we have to check if it has something with the grouping, and therefore we check if the relation has attribute class "Is inside", which means that *from* class is contained in the group. In this case we move the node from initial graph and add it to the graph of class *to* (line 15-18 4.2). In other cases, we create an edge and map its properties (line 20-26 4.2).

Algorithm 4.2: ADOxx2GraphML transformation.

Input: ADOxx model instance (XML file).
Output: GraphML model instance.

```
1  Graph g ← transformPackage(instance)
2  for Instance i : input.getElementsBy("INSTANCE") do
3      if getClassName(i) = "Grouping" then
4          Graph gGroup ← transformPackage(i)
5          g.add(gGroup)
6      else
7          Node n ← transformNode(i)
8          for Attribute a : i.getElementsBy("ATTRIBUTE") do
9              n.addAttribute(transformAttribute(a))
10         end
11         g.add(n)
12     end
13 end
14 for Connector c : input.getElementsBy("CONNECTOR") do
15     if c.getAttribute("class") = "Is inside" then
16         Graph gOrigin ← getGraph(c.to)
17         gOrigin.add(c.from)
18         g.remove(c.from)
19     else
20         Edge ed ← transformEdge(c)
21         ed.source ← c.from
22         ed.to ← c.to
23         for Attribute a : c.getElementsBy("ATTRIBUTE") do
24             ed.addAttribute(transformAttribute(a))
25         end
26         g.add(edge)
27     end
28 end
29 output.add(g)
30 return output
```

4.4 Papyrus to GraphML Transformation

This section shows the mapping from the Papyrus UML class diagram to GraphML. As shown in Table 4.3 we have specified only a few concepts. This is because we only focus on the UML class diagram and not all UML concepts. Here we have a bit different situation than before since both classes and relationships are represented as *Class* objects in UML. In the case of classes, we map the class object of type Class, AssociationClass, Package, Component, and DataType to a node. In the case of relationships, we are

mapping class objects of type Generalization, Realization, Abstraction, Dependency, and Realization to an edge.

Papyrus UML Concept	**GraphML Concept**	**Note**
UML Model	GraphML	General graph placeholder.
Package	Graph	Different packages apply to different graphs.
Class	Node	Class represents graph node. Following types are considered: *uml:Class* *uml:AssociationClass* *uml:Package* *uml:Component* *uml:DataType*
Class	Edge	Relation class represent graph edge. Following types are considered: *uml:Generalization* *uml:Realization* *uml:Abstraction* *uml:Dependency* *uml:Realization*
Class.Attribute	Node.Data	Applies for all nodes.

Table 4.3: PapyrusUML2GraphML transformation rules.

The pseudo-code can be seen in Algorithm 4.3. The implementation focuses on parsing the XML file where the UML model is stored. The algorithm consists of two parts. The first part is transforming the nodes, and the second is transforming the edges. XML structure of Papyrus UML stores all elements under a XML node called *packageImport*. Each of these XML nodes has a property called *xmi:type* where the name of class is stored (in some cases this property is called *xsi:type*). As we are dealing with UML models are considering the following types uml:Class, uml:AssociationClass, uml:Package, uml:Component, and uml:DataType. We iterate through all *packagedElement* nodes and we check if the type is one of the class name types (line 4-11 4.3). In the second part, where we have to transform edges, we follow the same procedure, but now we filter only XML elements that represent relationships (line 13-22 4.3). It is important to note here that there are differences in identifying an edge's source and target values. XML structure of Papyrus UML files stores this information differently for specific classes. The id of the source element is always stored in *xmi:id* property. In case of *uml:Generalization* it is stored in attribute called *general*, in case of *uml:Class* it is stored in attribute called *name*, in case of *uml:ElementImport* in attribute called *importedElement* and in other

Algorithm 4.3: PapyrusUML2GraphML transformation.

```
   Input: Papyrus model instance (XML file).
   Output: GraphML model instance.
 1 classes ← {"uml : Class", "uml : AssociationClass", "uml : Package", "uml :
     Component", "uml : DataType"}
 2 relations ← {"uml : Class", "uml : Abstraction", "uml : Dependency", "uml :
     Realization", "uml : Association"}
 3 Graph g ← transformPackage(instance)
 4 for Instance i : input.getElementsBy("packagedElement") do
 5     if classes.contains(i.getAttribute("xmi:type")) then
 6         Node n ← transformNode(i)
 7         for Attribute a : i.getAttributes() do
 8             n.addAttribute(transformAttribute(a))
 9         end
10         g.add(n)
11     end
12 end
13 for Instance i : input.getElementsBy("packagedElement") do
14     if relations.contains(i.getAttribute("xmi:type")) then
15         Edge ed ← transformEdge(i)
16         ed.source ← getObject(i.getAttribute("xmi : id"))
17         ed.to ← getObject(getTargetAttributeBasedOnClassType(i))
18         for Attribute a : i.getAttributes() do
19             g.addAttribute(transformAttribute(a))
20         end
21         g.add(edge)
22     end
23 end
24 output.add(g)
25 return output
```

cases in attribute called *type*. Therefore it is necessary to correctly identify the target element of an edge (line 17. 4.3).

4.5 Archi- And ArchiMate-Specific Transformation to GraphML

As shown in Fig 4.2 the transformation can include some domain-specific transformation steps. For example, the ArchiMate model can be developed upon different meta-models. One such example is Ecore, and also one such tool that follows this is Archi. An example of overriding the general rules is given in the next section.

Here we handle the domain-specific transformation. We will override generic transformation Ecore2GraphML (Table 4.1) to support the specific needs of some modeling language (i.e., ArchiMate). What is explicitly included in the Archi tool are the folders and diagram models. We handle this so that we add a rule that transforms a *Grouping*, *Folder*, or *DiagramModel* concept into a nested *Graph*. The initial transformation would yield a node for this. Also, what is important is to put all content associated with these concepts into nested groups so we can utilize possibilities provided by GraphML. If some entity were a part of a folder, it would be contained in a nested graph in GraphML. Additionally, since we are using the Ecore2GraphML general transformation, we need to exclude unnecessary concepts developed additionally in ArchiMate meta-model. One concept, for example, is *DiagramModelArchimateObject*, which is used to store the information about how the object is rendered in the Archi tool, which is not helpful information for analysis that will take place on a resulting graph, so no need to transform this.

ArchiMate (Archi)	**GraphML**	**Note**
Grouping IFolder IDiagramModel	Nested graph	Different groupings, folders, and diagrams correspond to different graphs.
IArchimateRelationship (iAR)	Edge	This specific Archi concept correspond to an edge. $iAR.parent \rightarrow Edge.Source$ $iAR.target\ To \rightarrow Edge.Target$
IFolder *IDiagramModel* *IDiagramModel* *DiagramModelBendpoint* *DiagramModelArchimateConn* *DiagramModelNote* *DiagramModelReference* *Bounds* *IArchimateRelationship* *ArchimateModel*	*no mapping*	This entities are excluded since they are not relevant.

Table 4.4: Archi- and ArchiMate-specific transformation rules.

In this domain-specific example, the different way of storing relationships in Archi is also important to consider. Relations are stored as entities *IArchimateRelationshipEntity*s, and in the default configuration, they will be transformed into nodes which is not correct (refer to Fig. 2.6). This has to be changed, and the IArchimateRelationshipEntity should be transformed into an *Edge*. This rule is also included in Tab. 4.4.

CHAPTER 5

Prototype Platform

This chapter is devoted to details about implementing a prototype platform for the framework introduced in section 4.1 described in Fig. 4.1). The platform is called Conceptual Model to Knowledge Graph Platform (CM2KG)[1]. Several things can be considered when it comes to creating a prototype platform for the introduced framework. These topics are related to storage, performance, user interface, and depending on this, different technologies can be used to achieve such requirements. The platform developed for this thesis is cloud-based, and the reason for this is to enable as much as possible that cloud solutions offer. Also, this chapter will contain information on how different centralities can be used and how different EA and UML Code smells queries can be implemented in a graph-based manner. The code for all this can be found in the GitHub repository [BS21].

5.1 Platform Overview

In this section, we will discuss the prototype platform architecture. Again, this can be implemented in a different way and is dependant on software requirements itself. The platform has three different parts, which can be seen in Fig. 5.1. These parts are *Modeling Tools*, *CM2KG Cloud*, and *Third-Party Tools*. It starts with the Modeling Tools, where users create models in different modeling tools. Next, models are then exported in standardized formats (mostly XML structured files). Previous step is then input for the next part in the architecture chain, i.e., *CM2KG Cloud*. This part is where the prototype platform is deployed and hosted. It consists of application and database servers running in two different virtual machines (it can also run on one machine only). The application server has the functionality to host the web application, which servers as a user interface, and additionally, it has logic to transform models. This web application is intended to be

[1]CM2KG platform [online]: `https://me.big.tuwien.ac.at/`

extended with different analysis modules like smell analysis, which will be discussed later. On the other side, the database server has the role of storing the transformed graph and provide different graph algorithm functionalities. In our case, for the background graph database, we will use the Neo4j database, which is very popular and capable of handling millions of nodes and edges, additionally providing SQL-like query syntax called Cypher. The web application will use the neovis.js [Con21] library to render and visualize query results from the neo4j database. Third and last part of the platform is the cooperation with third-party tools. These tools are used if further analysis is needed and the web interface cannot answer what users want. This is enabled by using the GraphML format and third-party tools [Bra+13] that support this format like Gephi or yEd.

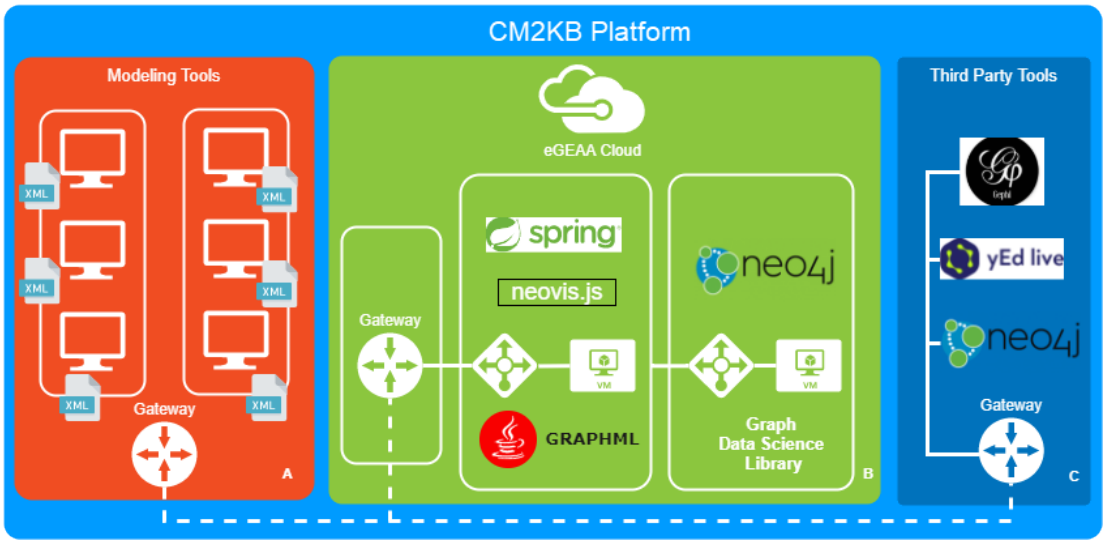


Figure 5.1: CM2KG platform architecture. [SHB21]

5.1.1 Process

In this section, we discuss what is required from the end-user (stakeholder that is doing the analysis) are discussed. From the user's perspective, the user must provide an instance of the model that is being analyzed in the structure explained in the next section. The next step is to add some custom configuration to the model transformation if necessary. The last step represents the output of the whole process, i.e., a GraphML file used to leverage the possibilities of graph theory.

5.1.2 Input

As discussed in the section for transforming enterprise architecture into graphs, the input is an instance of the enterprise architecture model, and depending on the implementation of such model specific model transformation is executed. In the prototype implementation

CM2KG uses two types of inputs: ArchiMate Model Exchange File Format [Lib19] and ADOxx ADL file.

Open Group ArchiMate Model Exchange File Format Standard represents a format intended to be a "standard for ArchiMate model exchange data between tools that need to import and export ArchiMate notation"[Lib19]. These "exchange files enable exporting content from one ArchiMate modeling tool or repository and importing it into another while retaining information describing the model in the file and how it is structured, such as a list of model elements and relationships"[Lib19]. The file itself uses standardized XML representation, and therefore it is easy to do the transformation of this model representation. The overall view of this XML file representation can be seen as a *model* node on top with some metadata nodes inside this node as *name* and *documentation*. Then the *elements* node follows having all the *element* nodes inside. The same applies to *relationships*.

```
<model xmlns="http://www.opengroup.org/xsd/archimate/3.0/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.opengroup.org/xsd/archimate/3.0/
    http://www.opengroup.org/xsd/archimate/3.1/archimate3_Model.xsd"
    identifier="model-1" version="1.0">
    <name xml:lang="en">Basic Model</name>
    <documentation xml:lang="en">Example of a basic model with
    two elements and a relationship</documentation>
    <elements>
        <element identifier="ba1" xsi:type="BusinessActor">
            <name xml:lang="en">A Business Actor</name>
        </element>
        <element identifier="br1" xsi:type="BusinessRole">
            <name xml:lang="en">A Business Role</name>
        </element>
    </elements>
    <relationships>
        <relationship identifier="relation-1" source="ba1"
        target="br1" xsi:type="Assignment">
            <name xml:lang="en">Assignment Relationship</name>
        </relationship>
    </relationships>
</model>
```

Listing 5.1: Archi example input file.

ADOxx XML ADL file represents the standardized model representation for the ADOxx platform described in XML notation. Because of the unified XML structure, like in ArchiMate Model Exchange File Format Standard, it is easy to parse and process the file following the transformation rules described in Section 4.3. On the high level, the model consists of a *models* node that can have multiple *model* nodes inside. Furthermore, each *model* has a *model attributes* node, which can have multiple *attribute* nodes. In the same *model*, node entities are stored in *instance*

nodes with multiple *attribute* nodes. The same applies to the *connector* which have inside the nodes *from* and *to*.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ADOXML SYSTEM "adoxml31.dtd">
<ADOXML version="3.1" ...>
    <MODELS>
        <MODEL id="mod.19809" name="test" version=""
        modeltype="Business Layer" libtype="bp"
        applib="...">
            <MODELATTRIBUTES>
                <ATTRIBUTE name="Version number"
                           type="STRING">
                </ATTRIBUTE>
            </MODELATTRIBUTES>
            <INSTANCE id="obj.19811" class="Grouping"
                      name="Grouping-19811">
                <ATTRIBUTE name="External tool coupling"
                           type="STRING">
                </ATTRIBUTE>...
            </INSTANCE>
            <INSTANCE id="obj.19822"
                      class="Business collaboration"
                      name="Business collaboration-19822">
                <ATTRIBUTE name="External tool coupling"
                type="STRING"></ATTRIBUTE>...
            </INSTANCE>
            <CONNECTOR id="con.19825" class="Is inside">
                <FROM instance="Business collaboration-19822"
                class="Business collaboration"></FROM>
                <TO instance="Grouping-19811"
                class="Grouping"></TO>
                <ATTRIBUTE name="AutoConnect"
                           type="STRING">
                </ATTRIBUTE>
            </CONNECTOR>
        </MODEL>
    </MODELS>
</ADOXML>
```

Listing 5.2: ADOxx example input file.

Papyrus UML file similar to the ADOxx XML file, Papyrus also packs files as XML unified structures. It consists of *uml:Model* as the initial node and core information as *packagedElement* nodes. Each node has an *id* and *type* as well as a *name*. An example of such a file is as follows.

```
<?xml version="1.0" encoding="UTF-8"?>
<uml:Model xmi:version="20131001"
           xmlns:xmi="http://www.omg.org/spec/XMI/20131001"
           xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore"
           xmlns:uml="http://www.eclipse.org/uml2/5.0.0/UML"
           xmi:id="_mRWEkOAoEeu-XNcrlfd6Uw" name="PapyrusTestModel">
```

```
7     <packageImport xmi:type="uml:PackageImport"
8                    xmi:id="_mU8LIOAoEeu-XNcrlfd6Uw">
9         <importedPackage xmi:type="uml:Model"
10                         href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes
      .library.uml#_0"/>
11    </packageImport>
12    <packagedElement xmi:type="uml:Class"
13                     xmi:id="_BK2jIOYHEeuzZOeHOQQcTg"
14                     name="Class2"/>
15    <packagedElement xmi:type="uml:Class"
16                     xmi:id="_BaMmsOYHEeuzZOeHOQQcTg"
17                     name="Class3"/>
18    <packagedElement xmi:type="uml:Abstraction"
19                     xmi:id="_CIblsOYHEeuzZOeHOQQcTg"
20                     name="someAbstraction"
21                     client="_BK2jIOYHEeuzZOeHOQQcTg"
22                     supplier="_BaMmsOYHEeuzZOeHOQQcTg"/>
23 </uml:Model>
```

Listing 5.3: ADOxx example input file.

5.1.3 Output

As discussed in Section2.3, GraphML format (XML file) is the main output of the whole process. This specific graph representation schema is used because it contains the most features and is supported by many different tools [Bra+13]. This specific format allows the enterprise architects to use different tools and leverage all the features graph analysis tool offers. Some of the features this file and the CM2KG platform provide are discussed in the next section.

5.1.4 Components

This section will provide details on the functionality that CM2KG cloud platform provides.

Model Transformation & Inspection. The first functionality is uploading an EA model (XML format or another standardized format for specific meta-modeling tools and transformation to a GraphML graph-structured format. Initial and transformed models can then be compared. The result of the transformation, the GraphML file, can then be downloaded if necessary for the user.

Graph Visualisation & Analysis. When the model is transformed, and the resulting graph is there, it is possible to visualize this graph which can be done directly in the browser. This visualization can be configured to set up different parameters for the size and color of the nodes and how the node label will be presented. This visualization itself automatically provides a tool for performing analysis on a model. Furthermore, this component provides a possibility for the user to select some of the predefined graph algorithms. These algorithms are based on graph centrality

and community detection metrics. Users can select an algorithm (and override it with different parameters if necessary) and perform analysis. These algorithms are provided directly by the graph database engine, which is the idea of the framework itself. The last and significant feature provided to a user in this component is the possibility of writing their neo4j query and executing it directly on the graph database. These queries are written in Cypher syntax and provide almost an unlimited source of possibilities to extract some information.

Interoperability. As the platform is running in the cloud and utilizes a graph database engine, it is possible to provide the resulting graph to other tools. Each transformed graph can be easily reached by URL, and this enables again connecting Neo4j Desktop Browser or some different graph analysis third-party tool directly. This way, the platform bridges the resulting graph and different tools by providing "a hook" to a standardized graph format to utilize different possibilities provided by different tools and libraries.

5.1.5 Transformation

The platform uses the implemented libraries to transform the models. Depending on the size of the model, performance can differ. To speed up transformation, some optimizations are implemented which gave better results. The transformation libraries are designed so that new transformations for other models are easily incorporated.

5.1.6 Web Interface

The web interface consists of several pages. In the following text, each page is described along with the screenshoot. Homepage of the platform[2] is shown in Fig. 5.2.

Model type selection. This is the starting page (see Fig. 5.3), and the user can select the model type he wants to upload. There are four possibilities to choose from: Ecore, Archi, ADOxx, and Papyrus UML . When a user clicks the button, he is redirected to the next page.

File selection. In this view (see Fig. 5.4), the user is asked to provide the input file or files he wants to transform. As said on the previous page, there were four possibilities. For example, if the second one is selected, i.e., Archi, the user is asked to upload the ArchiMate Model Exchange File Format. If the user selects for example third option, i.e., ADOxx, two files are required: ADOxx XML ADL file and .dtd file. After this is done, the user can proceed to the next screen.

Model comparison. In this screen (see Fig. 5.5), a user is presented with both models: the input and the transformed GraphML model. The reason for this is to give a direct comparison of the models so the user can check how the initial information

[2]CM2KG platform [online]: `https://me.big.tuwien.ac.at/`

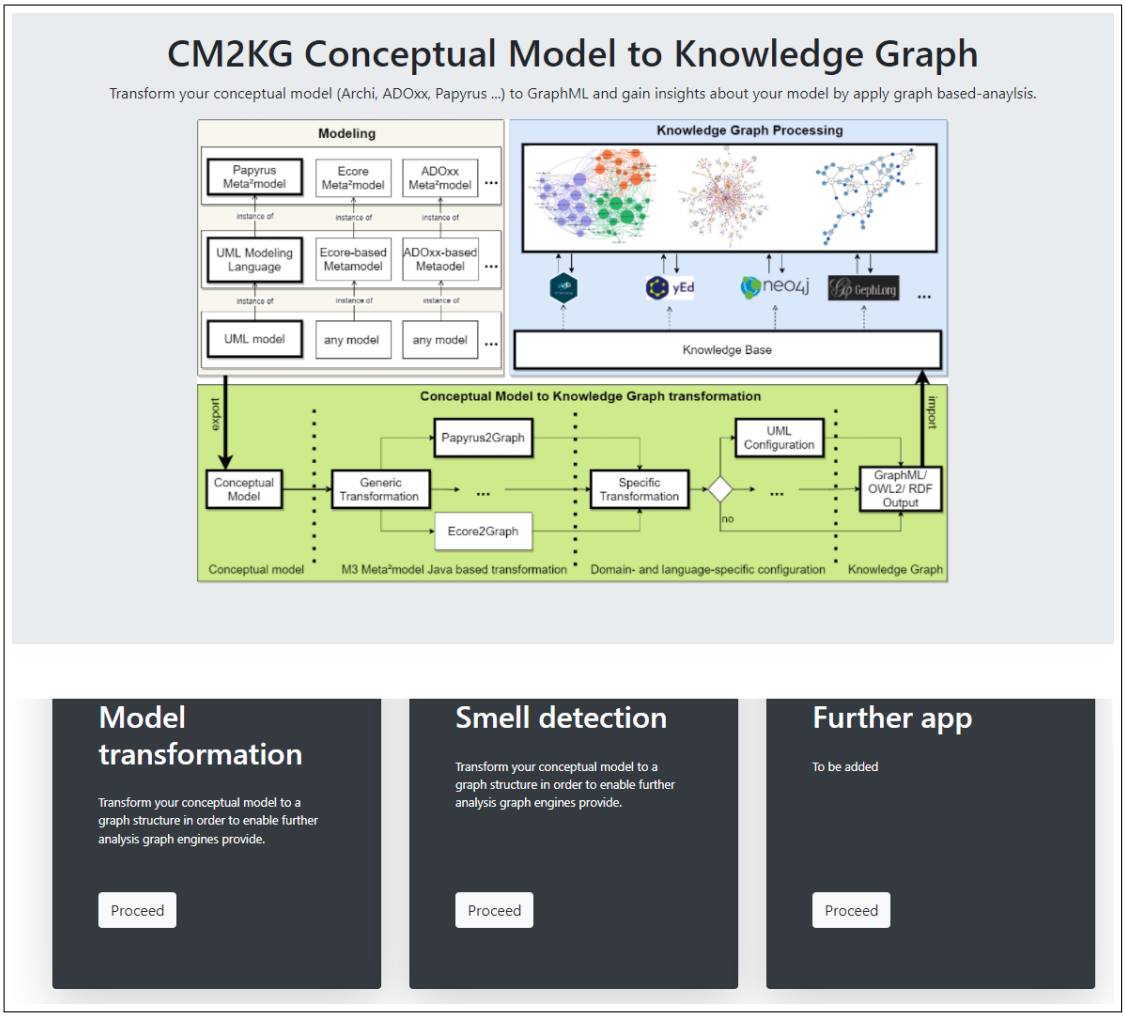


Figure 5.2: CM2KG home screen.

is transformed into the desired one. Multiple possibilities are possible from this screen. Users can either download the raw model. Users can open the raw XML preview of the GraphML model. Next, the user can download the GraphML file format. Lastly, user can open the open the model analysis view by initializing Neo4j graph.

XML Model preview. As said in the previous point, users can have a direct XML preview of the GraphML model (see Fig. 5.6).

Model Analysis This is the page that is based on Graph Science Playground Library for Neo4j (see Fig. 5.7). Users can do multiple things here. In the central part, the GraphML model is always visible. Additionally, users can type the Cypher query to query the graph. Next, the user can select some of the centrality detection and

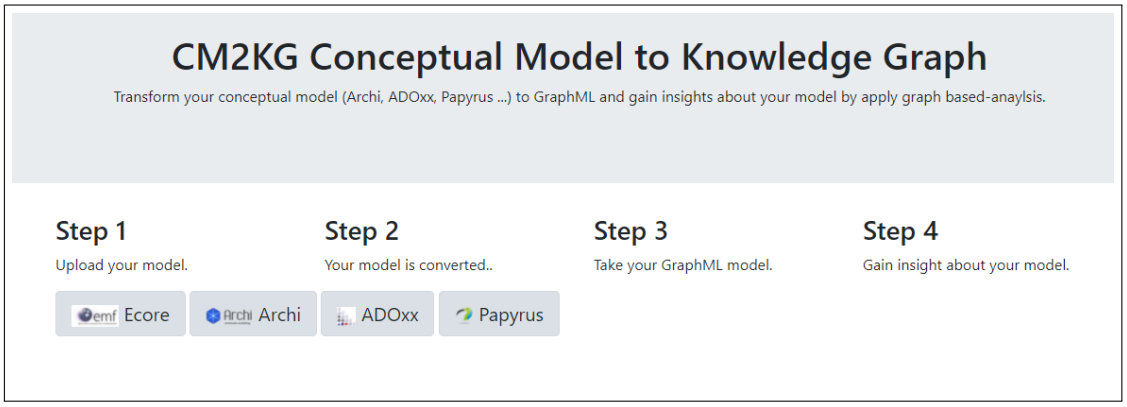


Figure 5.3: CM2KG model type selection.

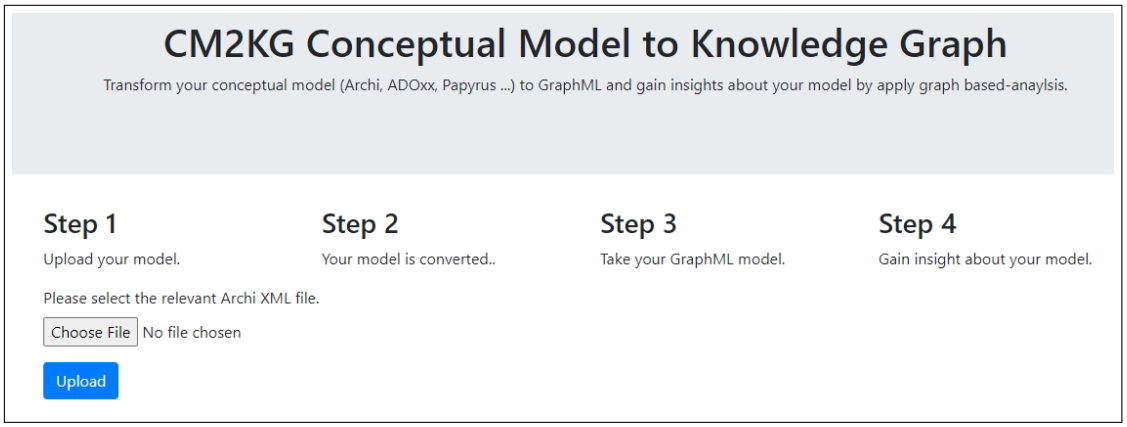


Figure 5.4: CM2KG *Open Group ArchiMate Model Exchange File Format* selection.

community detection algorithms and add some configuration in the algorithm query field. Users can also configure some things like which field in the GraphML will be used to render the node's size and which field should be used to specify the caption for a node. Each change in the configuration will refresh the graph preview.

5.1.7 External Tools

Here we discuss the intention to use standardized formats like GraphML supported by tools like Neo4j, yED, and Gephi. Here, we present some possibilities that this concept is enabling. Gephi supports large graph visualizations. It enables the execution of different graph centrality algorithms, provides different visualizations for nodes and edges in different colors and sizes. Additionally, it provides different pre-made layouts for graph, which ends up in many possibilities to analyze a graph. yED is another tool, and it provides similar possibilities as Gephi, but it also provides more choices for configuring the layout of nodes and relations. On the other side, neo4j is a graph database engine

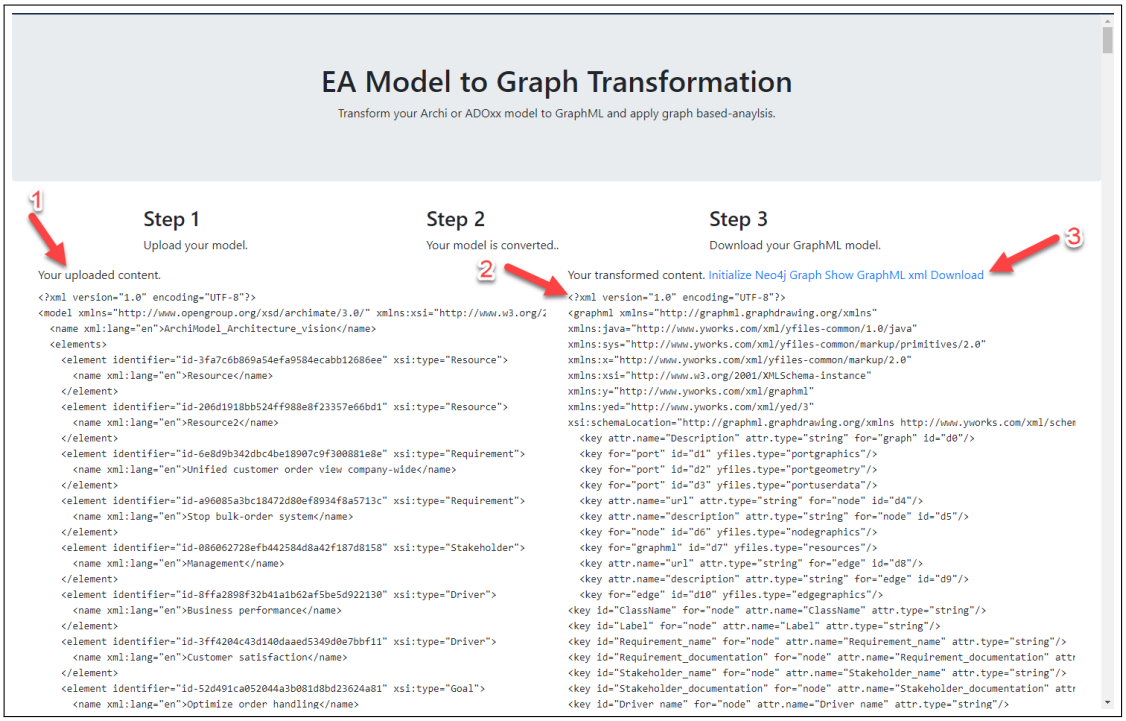


Figure 5.5: CM2KG Model comparison.

capable of handling millions of nodes and edges. It provides many different algorithms, and most importantly, it provides querying in SQL-like syntax called Cypher.

5.2 Use of Graph Metrics

This section will provide examples of some graph metrics that can be used to analyze a graph. Table. 5.1 shows potential questions and metrics and how they can be utilized in the analysis of Enterprise Architecture. This table was created following the goal question metric approach as the basis for formulating enterprise architect competency questions.

Summarized and more EA analysis-oriented questions can be found in Table. 5.2. This table represents example enterprise architect questions and a graph metric that could be used to answer the question.

5.2.1 Centralities

This section will show examples of how the centralities can be used with respect to Enterprise Architecture. All Neo4j queries in this section are written using Graph Data Science Library (GDS) version 1.4.0 and APOC 4.1.0.6 neo4j plugins.

Type	Number	Example
Goal	Purpose Issue Object Viewpoint	Improve Structural analysis of Dependencies in EA models From EA architect viewpoint
Question	Q1.1	Which are the most important elements?
Metric	M1.1.1 M2 M3 M4 M5	Degree Page Rank Closeness centrality Betweenness centrality Eigenvector centrality
Question	Q1.2	If X is down which Y is affected the most?
Metric	M1.2.1 M7	Baseline importance / New importance Modularity
Question	Q1.3	What is the current speed of providing answers?
Metric	M1.3.1	Average processing time
Question	Q1.4	Is the performance Satisfactory from EA Architect viewpoint?
Metric	M1.4.1	Subjective evaluation by the project manager
Question	Q1.5	Is the speed of providing answers improving?
Metric	M1.5.1 M10	Current time / baseline time Subjective evaluation by the project manager
Goal	Purpose Issue Object Viewpoint	Improve Structural analysis of Coverage in EA models From EA architect viewpoint
Question	Q2.1	Is X affected by a change in Y?
Metric	M2.1 M2.1.2	Baseline importance / New importance Modularity
Question	Q2.2	If X is down which Y is affected the most?
Metric	M2.2.1	Baseline importance / New importance
Question	Q2.3	What is the current speed of providing answers?
Metric	M2.3.1	Average processing time
Question	Q2.4	Is the performance Satisfactory from EA Architect viewpoint?
Metric	M2.4.1	Subjective evaluation by the project manager
Question	Q2.5	Is the speed of providing answers improving?
Metric	M2.5.1 M2.5.2	Current time / baseline time Subjective evaluation by the project manager
Goal	Purpose Issue Object Viewpoint	Improve Structural analysis of Complexity in EA models From EA architect viewpoint
Question	Q3.1	Which are the most complex elements?
Metric	M3.1.1	Degree
Question	Q3.2	What is the current speed of providing answers?
Metric	M3.2.1	Average processing time
Question	Q3.3	Is the performance Satisfactory from EA Architect viewpoint?
Metric	M3.3.1	Subjective evaluation by the project manager
Question	Q3.4	Is the speed of providing answers improving?
Metric	M3.4.1 M3.4.2	Current time / baseline time Subjective evaluation by the project manager

Table 5.1: Goal question metric approach in enterprise architecture proposal.

Figure 5.6: CM2KG XML Model preview.

Graph metric	EA interpretation and exemplary competency question (CQ)
Centralities	
Degree	The higher the value the more edges a node has. CQ: How many application components are used by one business role?
Closeness	How close is a Node to the other graph components. CQ: What is the closest switch that can be used to connect two servers?
Betweenness	How important is a Node in connecting different parts of a graph? CQ: What is the impact of removing an application component?
Eigenvector	How important are the connected Nodes of a Node? CQ: What is the ranking of Business Actors?
Page Rank	How important is a Node for its directed network. CQ: What is the ranking of the most important Drivers in a given model?
Community Detection	
Weakly Connected Components	For one community there exists a path from each node to another one without considering the direction of the relationship. CQ: Is the device part of a network?
Strongly Connected Components	For one community every node is reachable from every other node when considering the direction of the relationship. CQ: Can each device in a group exchange information with another one?
Local Clustering Coefficient	Represents the likelihood that the neighbors are also connected. CQ: If the switch is down can neighbor switches help and overtake the traffic?

Table 5.2: Interpretation of graph metrics for ArchiMate models. [SB21b]

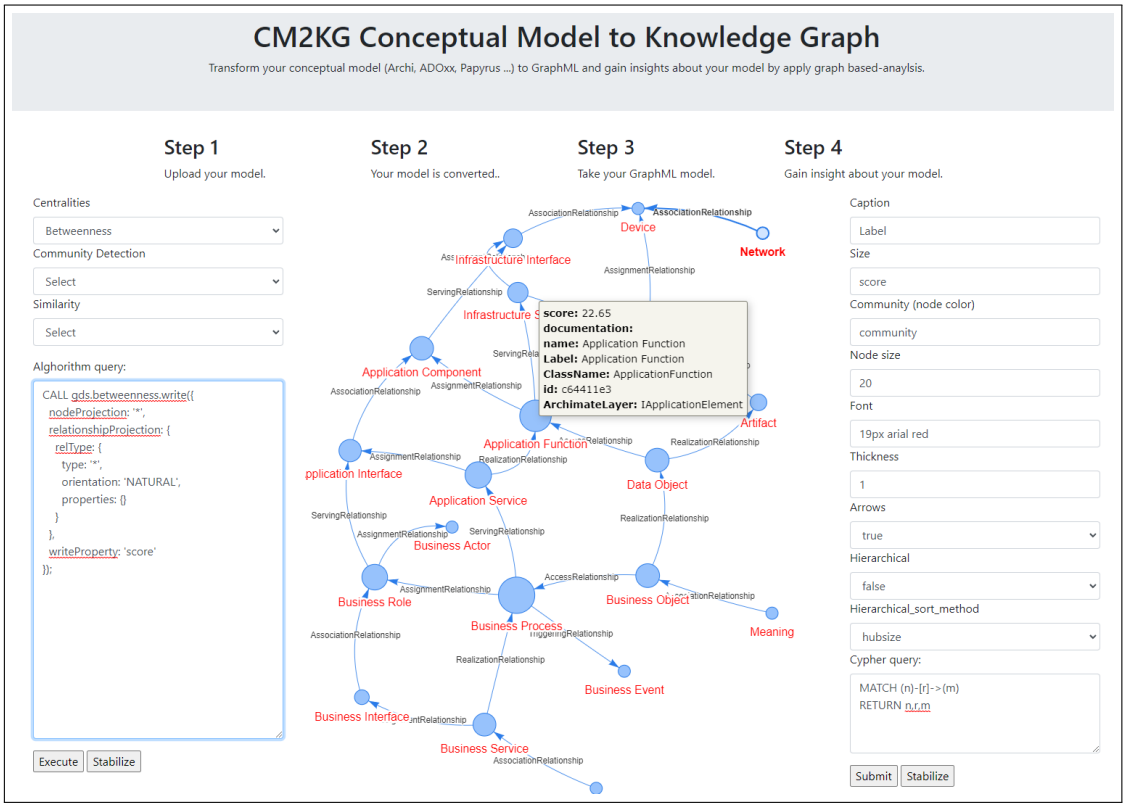


Figure 5.7: CM2KG graph-based model visualisation and analysis.

Degree

In this example, the ArchiMetal model from Archi is used as an example. In graph theory, the degree is related to how many edges a node has. There can be a differentiation between incoming or outgoing degrees depending on the type of edge if it is directed.
EA Question : *How many application components are used by one business role?*

In order to answer this question, a projection of the initial graph can be taken with the following nodes: *Business Role*, *Application Component*, and edges: *Serving Relationship*.

```
CALL gds.graph.create(
  'degree-subgraph',
  {
    BusinessRole: {},
    ApplicationComponent: {}
  },
  {
    ServingRelationship: {}
  }
)
```

Listing 5.4: Neo4j degree centrality projection preparation query.

After the projection is created, an algorithm calculating degree centrality can be executed.

```
CALL gds.alpha.degree.stream('degree-subgraph')
YIELD nodeId, score
with gds.util.asNode(nodeId) as n, score
where n.name="Some business role name"
return n.name, score
ORDER BY score DESC
```

Listing 5.5: Neo4j degree centrality result query.

It is important to note that the same degree centrality calculation could be done on the whole graph, but it is important to understand the context in which the degree is being evaluated and the business needs.

Closeness

In graph theory, closeness centrality can be described with the following question: *how close is a node to the other nodes*?
EA Question : *What is the closest switch that can be used to connect two servers?*

In order to answer this question, a projection of the initial graph can be taken with the following nodes: *Device*, *Communication Network*, and *Node* and relations: *Association Relationship* and *Realization Relationship*. Again it is important to note that we are projecting of the whole repository to the subset of nodes that would be interesting to our specific question.

```
CALL gds.graph.create(
  'closeness-subgraph',
  {
    Device: {},
    CommunicationNetwork: {},
    Node: {}
  },
  {
    AssociationRelationship: {},
    RealizationRelationship: {},
  }
)
```

Listing 5.6: Neo4j closeness centrality projection preparation query.

After the projection of the whole repository is created, we can then execute the algorithm for closeness centrality and filter the switch with the biggest closeness centrality.

```
CALL gds.alpha.closeness.stream('closeness-subgraph')
YIELD nodeId, score
with gds.util.asNode(nodeId) as n, score
where n.name contains "Switch"
return n.name, score
ORDER BY score DESC
```

Listing 5.7: Neo4j closeness centrality result query.

Betweenness

The same ArchiMetal model is used in this example. In graph theory, the betweenness is related to the importance of connecting two other nodes. Moreover, if the node is contained in the shortest path between two nodes, it will boost the betweenness centrality score.
EA Question : *What is the impact of removing an application component?*

In order to answer this question, a projection of the initial graph can be taken with the following nodes: *Application Component* and edges: *Flow Relationship*. The query is as follows:

```
CALL gds.graph.create(
  'betweenness-subgraph',
  {
    ApplicationComponent: {}
  },
  {
    FlowRelationship: {}
  }
)
```

Listing 5.8: Neo4j betweenness centrality projection preparation query.

After the projection is created, an algorithm for calculating betweenness centrality can be executed, and we can see the ordered list of application components scored by how important are they when standing in between others.

```
CALL gds.betweenness.stream('betweenness-subgraph')
YIELD nodeId, score
with gds.util.asNode(nodeId) as n, score
RETURN n.name, score
ORDER BY score DESC
```

Listing 5.9: Neo4j betweenness query result.

Eigenvector

In graph theory, eigenvector centrality can be described with the following question: *how important are the connected nodes of a node*?
EA Question : *What is the ranking of Business Actors?*

"Eigenvector Centrality is an algorithm that measures the transitive influence of nodes. A high eigenvector score means that a node is connected to many nodes who themselves have high scores"[neo21a]. To answer this question, a projection of all nodes and edges can be taken into consideration. The query is as follows:

```
CALL gds.graph.create(
  'eigenvector-subgraph',
  "*",
  "*"
)
```

Listing 5.10: Neo4j eigenvector centrality projection preparation query.

After the projection is created, an algorithm for calculating Eigenvector centrality can be executed.

```
CALL gds.eigenvector.stream('eigenvector-subgraph')
YIELD nodeId, score
with gds.util.asNode(nodeId) as n, score
WHERE n.ClassName = "BusinessActor"
return n.name, score
ORDER BY score DESC
```

Listing 5.11: Neo4j degree pagerank result query.

Page Rank

In graph theory, page rank centrality can be described with the following question: *how important is a node for its directed network*?
EA Question : *What is the ranking of most important Drivers in a given model?*

In graph theory, the PageRank is a variant of Eigenvector that works on directed networks where Eigenvector works with undirected networks.

"The PageRank algorithm measures the importance of each node within the graph, based on the number of incoming relationships and the importance of the corresponding source nodes. The underlying assumption roughly speaking is that a page is only as important as the pages that link to it." [neo21b]. In order to answer this question, a projection of all nodes and edges can be taken into consideration. The query is as follows:

```
CALL gds.graph.create(
  'pageRank-subgraph',
  "*",
  "*"
)
```

Listing 5.12: Neo4j pagerank centrality projection preparation query.

After the projection is created, an algorithm for calculating PageRank centrality can be executed.

```
CALL gds.pageRank.stream('pageRank-subgraph')
YIELD nodeId, score
with gds.util.asNode(nodeId) as n, score
WHERE n.ClassName = "Driver"
return n.name, score
ORDER BY score DESC
```

Listing 5.13: Neo4j degree pagerank result query.

5.2.2 Community Detection

Weakly Connected Components

For one community there exists a path from each node to another one without considering the direction of the relationship.
EA Question : *Is the device part of a network?*

In order to answer this question, a projection of the initial graph can be taken with the following nodes: *Device*, *Communication Network*, and *Node* and relations: *Association Relationship* and *Realization Relationship*.

```
CALL gds.graph.create(
  'wcc-subgraph',
  {
    Device: {},
    CommunicationNetwork: {},
    Node: {}
  },
  {
    AssociationRelationship: {},
    RealizationRelationship: {},
  }
)
```

Listing 5.14: Neo4j weakly connected components centrality projection preparation query.

After the projection of the whole repository is created, we can then execute an algorithm for community detection and show to which group the device belongs.

```
CALL gds.wcc.stream('wcc-subgraph')
YIELD nodeId, score
with gds.util.asNode(nodeId) as n, score
where n.name contains "Specific device name"
return n.name, score
ORDER BY score DESC
```

Listing 5.15: Neo4j weakly connected components centrality result query.

Strongly Connected Components

For one community, every node is reachable from every other node when considering the direction of the relationship.

EA Question : *Can each device in a group exchange information with another one?*

In order to answer this question, a projection of the initial graph can be taken with the following nodes: *Device*, *Communication Network*, and *Node* and relations: *Association Relationship*, *Realization Relationship*.

```
CALL gds.graph.create(
  'scc-subgraph',
  {
    Device: {},
    CommunicationNetwork: {},
    Node: {}
  },
  {
    AssociationRelationship: {},
    RealizationRelationship: {},
  }
)
```

Listing 5.16: Neo4j strongly connected components centrality projection preparation query.

After the projection of the whole repository is created, we can then execute an algorithm for the strongly connected components community detection and show all groups where each node can reach another node.

```
CALL gds.wcc.stream('scc-subgraph')
YIELD nodeId, score
with gds.util.asNode(nodeId) as n, score
where n.name contains "Specific device name"
return n.name, score
ORDER BY score DESC
```

Listing 5.17: Neo4j strongly connected components centrality result query.

Local Clustering

Represents the likelihood that the neighbors are also connected.
EA Question : *If the switch is down can neighbor switches help and overtake the traffic?*

In order to answer this question, a projection of the initial graph can be taken with the following nodes: *Node* and relations: *Association Relationship*. We want to check if a switch is connected to another two switches, and if it goes down, can two other switches still work, i.e., is there a connection between them.

```
CALL gds.graph.create(
  'lc-subgraph',
  {
    Node: {}
  },
  {
    AssociationRelationship: {}
  }
)
```

Listing 5.18: Neo4j local clustering centrality projection preparation query.

After the projection of the whole repository is created, we can then execute an algorithm for local clustering and show all nodes and their score concerning local clustering, showing if the neighbors of the selected node are connected.

```
CALL gds.localClusteringCoefficient.stream('lc-subgraph')
YIELD nodeId, score
with gds.util.asNode(nodeId) as n, score
return n.name, score
ORDER BY score DESC
```

Listing 5.19: Neo4j local clustering centrality result query.

5.3 EA Smell Detection

This section will introduce queries that enable finding the EA smell inside a model.

Chatty Service

A high number of operations is required to complete one abstraction. Such operations are typically rather simple tasks that needlessly slow down an entire process.

```
MATCH (a)-[r]-(b)
WHERE  a.ClassName contains 'Service' and b.ClassName contains 'Service'
WITH a,count(r) as cnt
WHERE cnt>4
MATCH (a)-[r1]-(b1)
WHERE  a.ClassName contains 'Service' and b1.ClassName contains 'Service'
RETURN a,b1, cnt
```

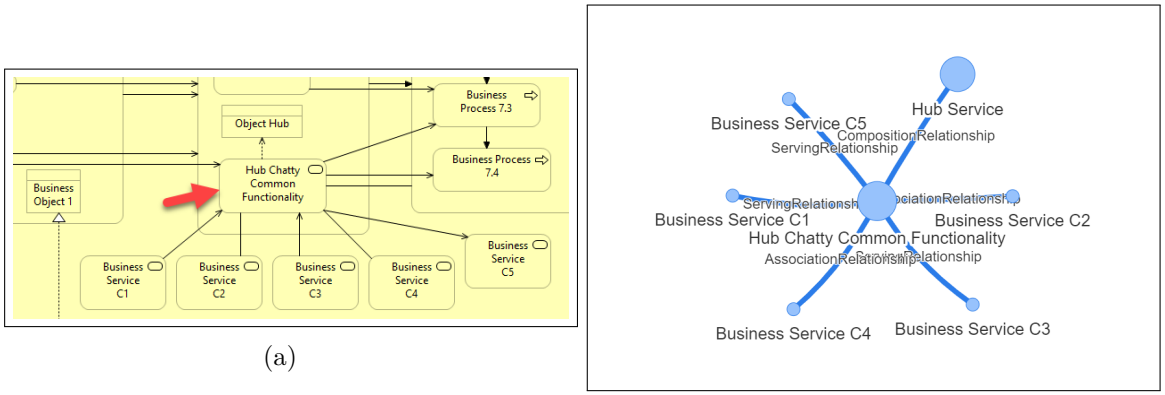
Listing 5.20: Cypher query to identify EA chatty service smell.

Cyclic Dependency

Two or more abstractions directly or indirectly depend on each other.

```
MATCH (a)-[r1]->(b)-[r2]->(c)-[]->(a)
RETURN a,b,c
```

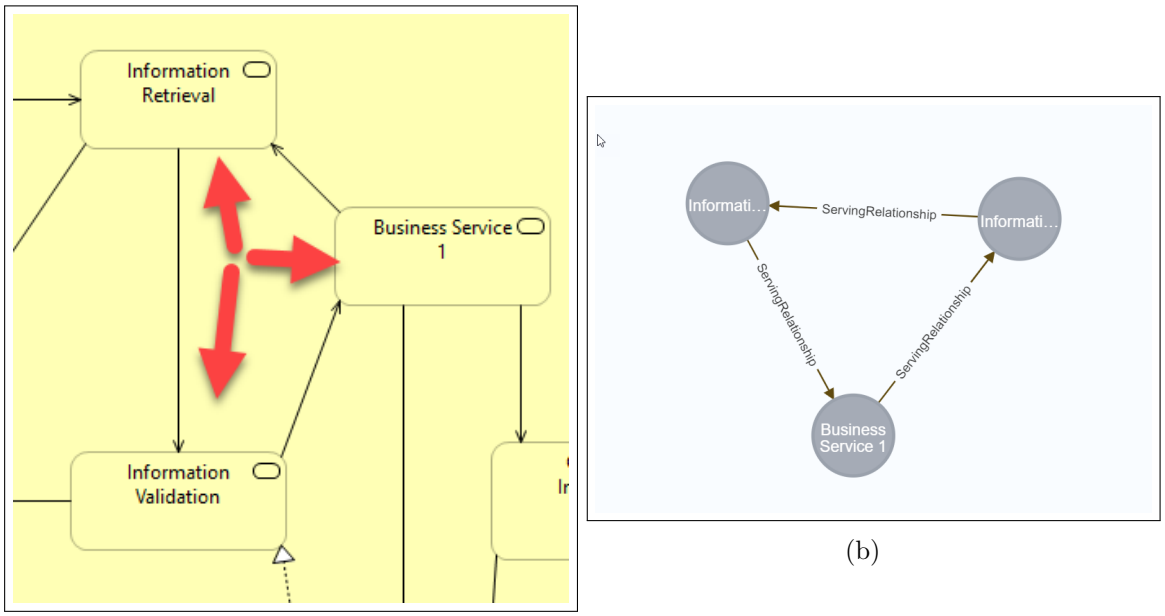
Listing 5.21: Cypher query to identify EA cyclic dependency smell.



(a)

(b)

Figure 5.8: Chatty service smell detection initial (a), the result of detection (b).



(a)

(b)

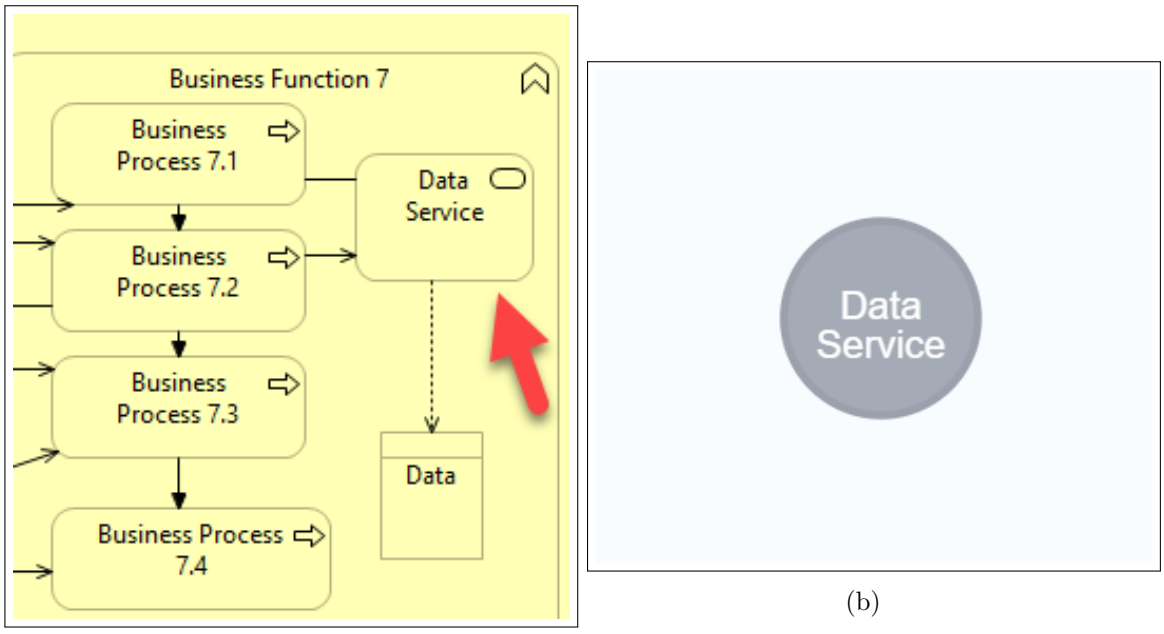
Figure 5.9: Cyclic Dependency smell detection initial (a), the result of detection (b).

Data Service

A service that exclusively performs information retrieval and typically provides only simple read operations.

```
MATCH (a)-[r1]-(b1)
WHERE a.ClassName='BusinessService' and (b1.ClassName = 'BusinessObject'
or b1.ClassName = 'DataObject' or b1.ClassName = 'SystemSoftware')
WITH a,r1,b1
MATCH (a)
WHERE not (a)--(:BusinessService)
RETURN a
```

Listing 5.22: Cypher query to identify EA data service smell.



(a) (b)

Figure 5.10: Data Service smell detection initial (a), a result of detection (b).

Dead Component

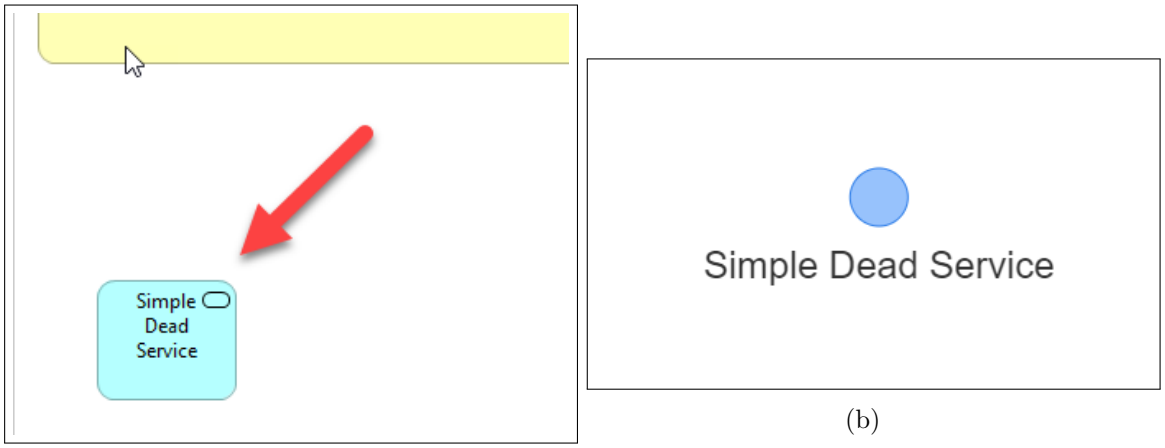
A component is no longer used or used to support potential future behavior.

```
MATCH (n)
WHERE not (n)--()
RETURN n
```

Listing 5.23: Cypher query to identify EA dead component smell.

Dense Structure

An EA repository has dense dependencies without any particular structure.

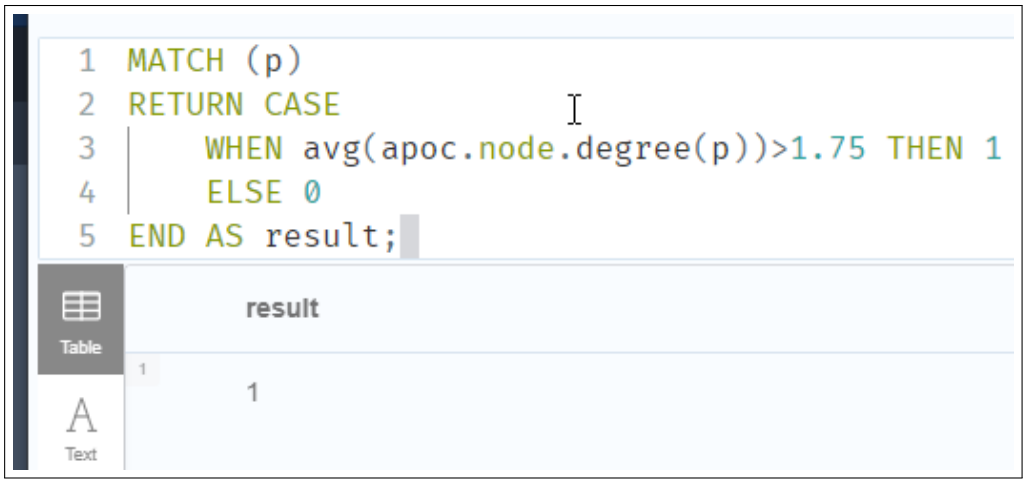


(a) (b)

Figure 5.11: Dead Component smell detection initial (a), the result of detection (b).

```
MATCH (p)
RETURN CASE
    WHEN avg(apoc.node.degree(p))>1.75 THEN 1
    ELSE 0
END AS result;
```

Listing 5.24: Cypher query to identify EA dense structure smell.



(a)

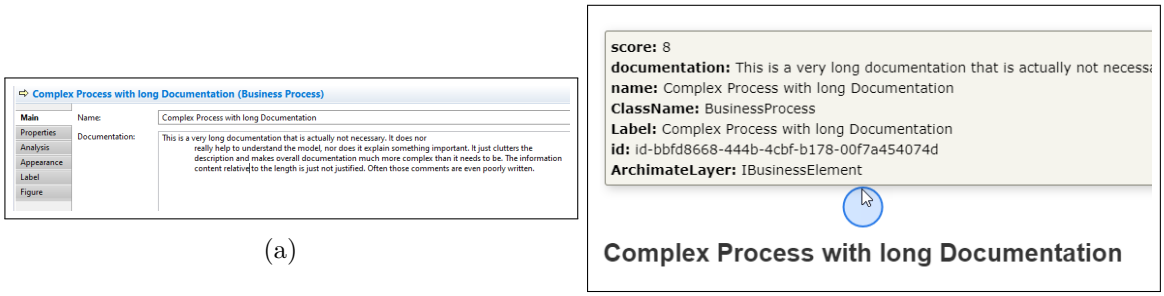
Figure 5.12: Dense Structure smell detection result (true/false).

Documentation

Lengthy documentation often points to unnecessary complex structures.

```
MATCH (n)
WHERE size(n.documentation)>256
RETURN n
```

Listing 5.25: Cypher query to identify EA documentation smell.



(a)

(b)

Figure 5.13: Documentation smell detection initial (a), a result of detection (b).

Duplication

Two or more abstractions with highly similar functionality exists.

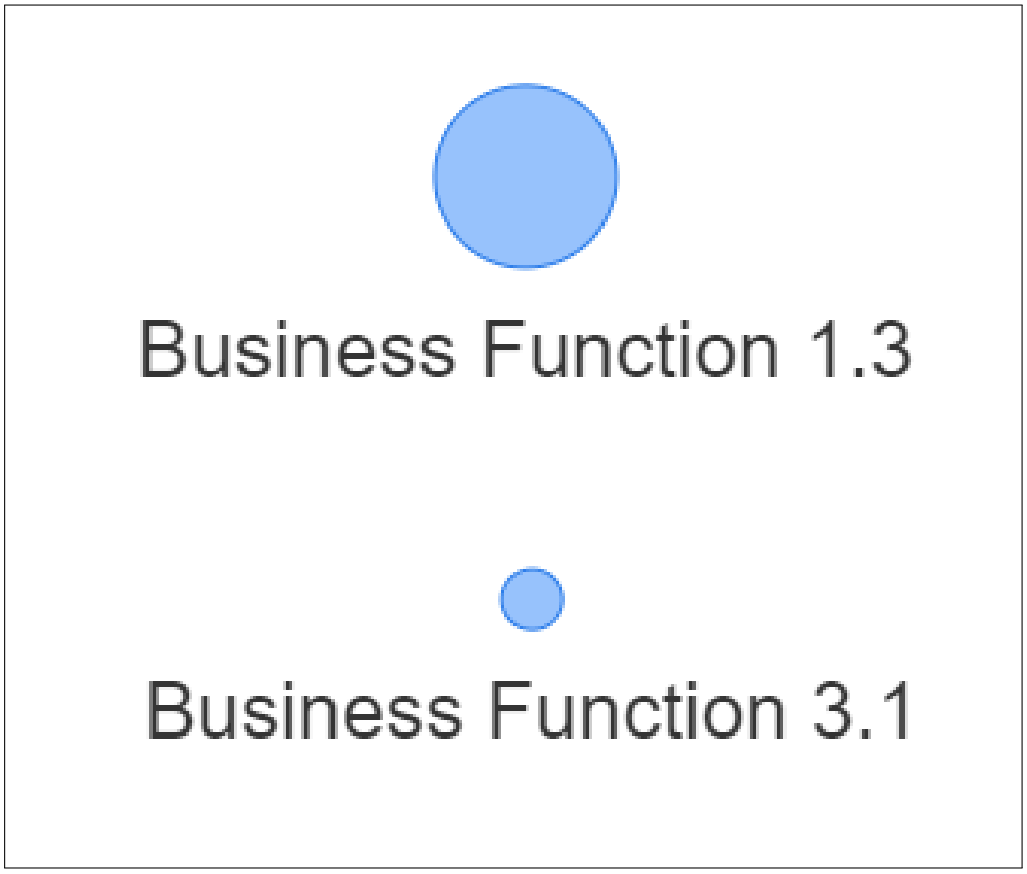
```
MATCH (a),(b)
WHERE a<>b and  a.ClassName = b.ClassName
    and apoc.text.jaroWinklerDistance(a.Label, b.Label)>0.8
RETURN a,b,
       apoc.text.jaroWinklerDistance(a.Label, b.Label) as similarNameScore
```

Listing 5.26: Cypher query to identify EA duplication smell.

Hub-like Modularization

This smell arises when an abstraction has dependencies (both incoming and outgoing) with a large number of other abstractions, being a single point of failure.

```
MATCH (a)-[r]-(b)
WHERE  (r.Label contains 'Aggregation'
        or r.Label contains 'Realization'
        or r.Label contains "Composition"
        or r.Label contains "Assignment")
        and a.ArchimateLayer = b.ArchimateLayer
WITH a, collect(r) as rels,  a+collect(b) as cluster
MATCH (m)-[r1]-(n)
WHERE not (r1.Label contains 'Aggregation'
    or r1.Label contains 'Realization'
    or r1.Label contains "Composition"
```

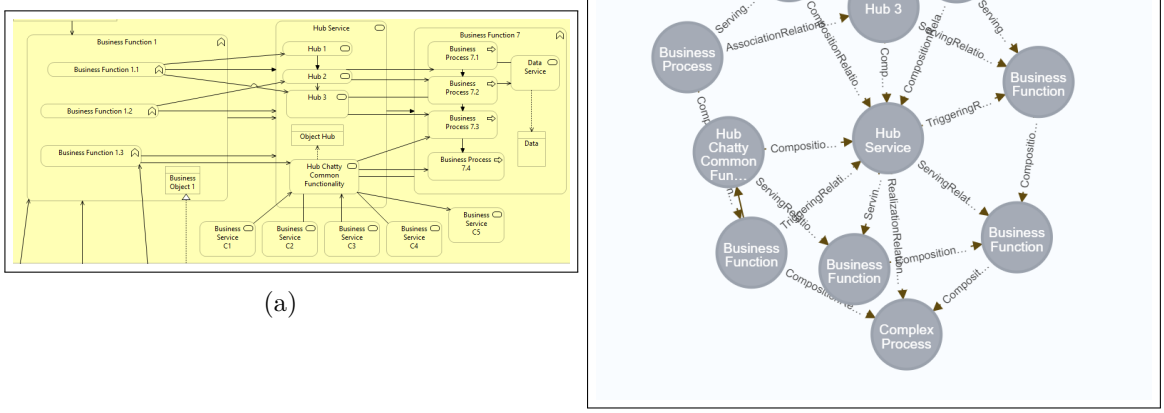


(a)

Figure 5.14: Documentation smell detection initial (a), a result of detection (b).

```
12     or r1.Label contains "Assignment") and
13 (m in cluster and not n in cluster)
14 WITH a, cluster, collect(r1) as fanout
15 MATCH  (m)-[r2]-(n)
16 WHERE not (r2.Label contains 'Aggregation'
17     or r2.Label contains 'Realization'
18     or r2.Label contains "Composition"
19     or r2.Label contains "Assignment") and
20     (not m in cluster and n in cluster)
21 WITH a, cluster,  fanout, collect(r2) as fanin
22 WHERE size(fanout) > 7 and size(fanin)>7
23 RETURN a, cluster, size(fanout), size(fanin)
```

Listing 5.27: Cypher query to identify EA hub-like modularization smell.



(a) (b)

Figure 5.15: Hub-like Modularization smell detection initial (a), a result of detection (b).

Lazy Component

A component that is not doing enough to pay for itself should be eliminated. Those components often only pass messages on to another.

```
MATCH (n)
WHERE n.Label contains 'controller'
      or n.Label contains 'manager'
RETURN n
```

Listing 5.28: Cypher query to identify EA lazy component smell.

Message Chain

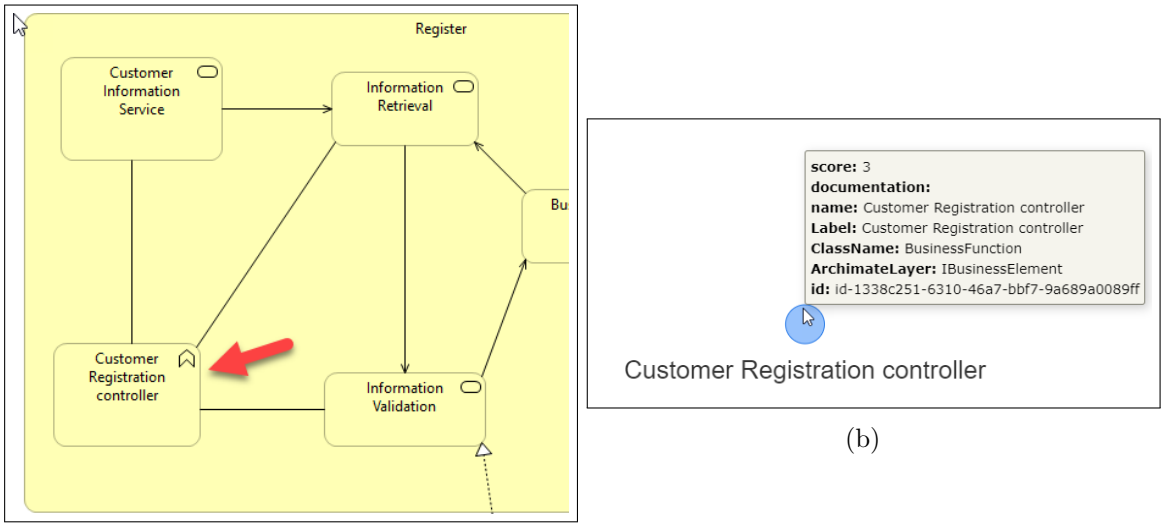
A chain of service calls and messages fulfills common functionality.

```
MATCH (a)-[r1]->(b)-[r2]->(c)-[]->(d)-[]->(e)
WHERE a.ClassName='BusinessService'
    and b.ClassName='BusinessService'
    and c.ClassName='BusinessService'
    and d.ClassName='BusinessService'
    and e.ClassName='BusinessService'
RETURN a,b,c,d,e
```

Listing 5.29: Cypher query to identify EA message chain smell.

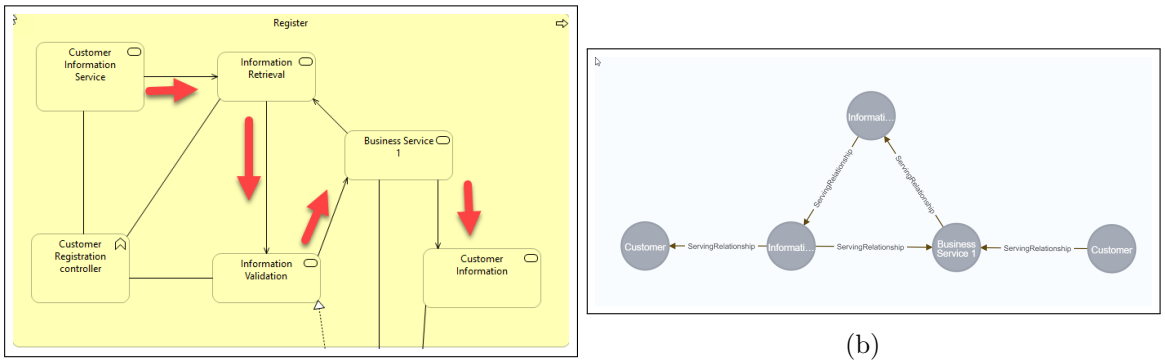
Shared Persistency

Different services access the same database. In the worst case, different services access the same entities of the same schema.



(a) (b)

Figure 5.16: Lazy Component smell detection initial (a), a result of detection (b).

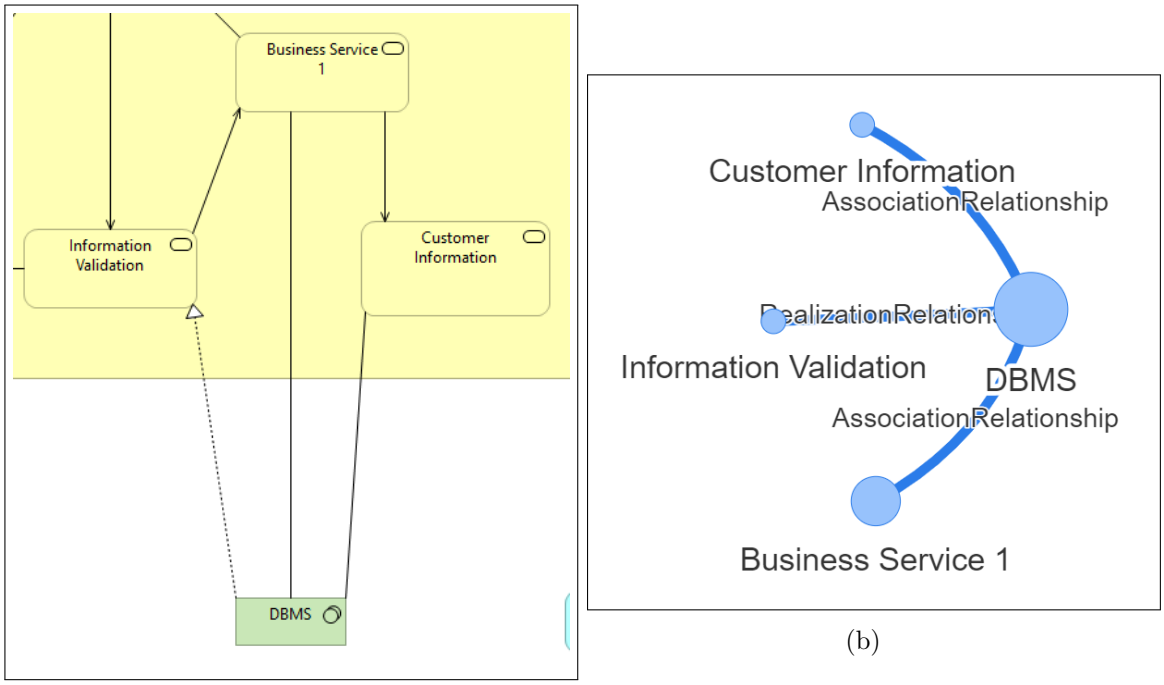


(a) (b)

Figure 5.17: Message Chain smell detection initial (a), a result of detection (b).

```
MATCH (a)-[r]-(b)
WHERE a.ClassName='SystemSoftware'
      and (r.Label='AssociationRelationship'
           or r.Label='RealizationRelationship'
           or r.Label='AssignmentRelationship')
WITH a,count(r) as cnt
MATCH (a)-[r1]-(b1)
WHERE cnt>1 and
      (r1.Label='AssociationRelationship'
      or r1.Label='RealizationRelationship'
      or r1.Label='AssignmentRelationship')
RETURN a,b1
```

Listing 5.30: Cypher query to identify EA shared persistency smell.



(a) (b)

Figure 5.18: Shared Persistency smell detection initial (a), a result of detection (b).

Strict Layers Violation

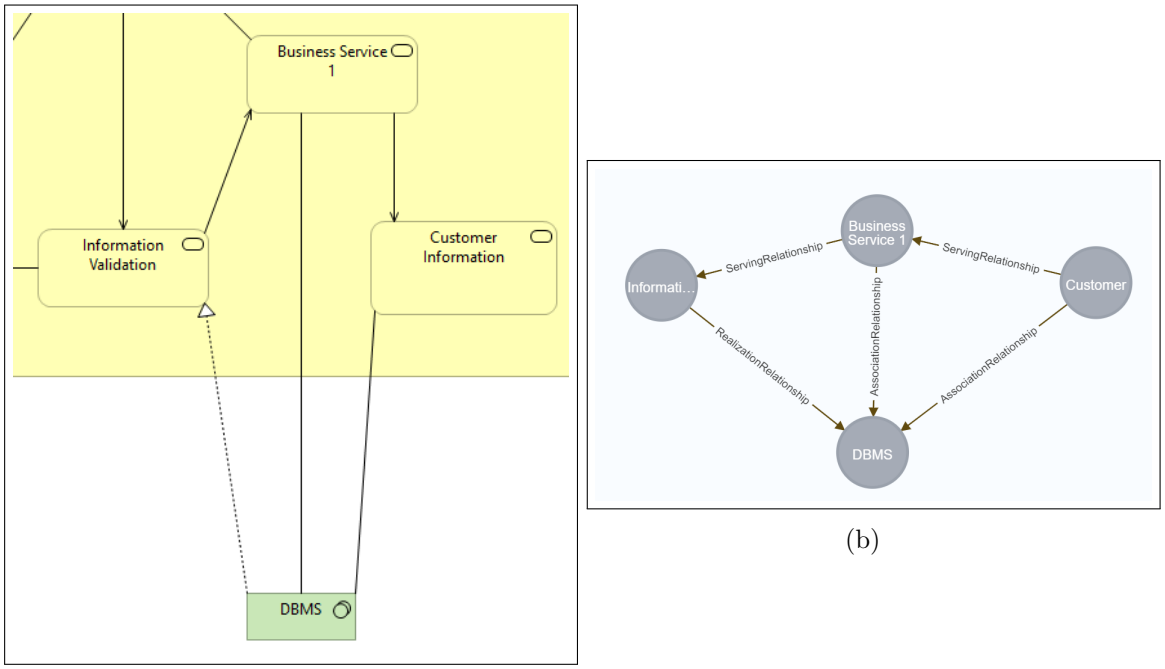
An element skips the EA layer directly beneath and accesses a layer further below instead.

```
1 MATCH (a)-[r]-(b)
2 WHERE a.ArchimateLayer contains 'Business'
3       and b.ArchimateLayer contains 'Technology' //example
4 RETURN a,b,r
```

Listing 5.31: Cypher query to identify EA strict layers violation smell.

Weakened Modularity

Each module must strive for high cohesion and low coupling. This smell arises when a module exhibits high coupling and low cohesion.



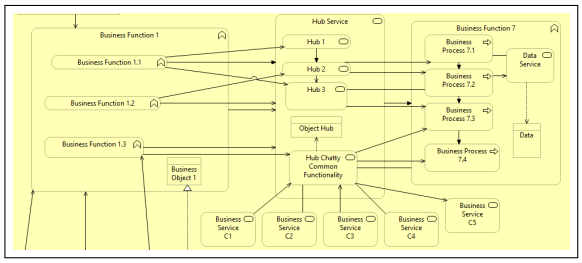
(a)

(b)

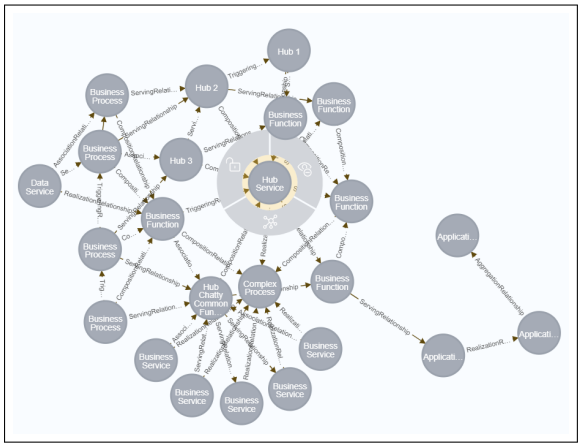
Figure 5.19: Strict Layers Violation smell detection initial (a), a result of detection (b).

```
MATCH (a)-[r]-(b)
WHERE  (r.Label contains 'Aggregation'
    or r.Label contains 'Realization'
    or r.Label contains "Composition"
    or r.Label contains "Assignment")
    and a.ArchimateLayer = b.ArchimateLayer
WITH a, collect(r) as rels,  a+collect(b) as cluster
MATCH (m)-[r1]-(n)
WHERE m in cluster and n in cluster
WITH a, cluster, collect(r1) as internal
MATCH  (m)-[r2]-(n)
WHERE not (r2.Label contains 'Aggregation'
    or r2.Label contains 'Realization'
    or r2.Label contains "Composition"
    or r2.Label contains "Assignment") and
    (not m in cluster and n in cluster)
    or (m in cluster and not n in cluster)
WITH a, cluster,  internal, collect(r2) as external
WHERE size(internal) < size(external) and size(internal)>3
RETURN a, cluster, size(internal), size(external)
```

Listing 5.32: Cypher query to identify EA weakened modularity smell.



(a)



(b)

Figure 5.20: Weakened Modularity smell detection initial (a), a result of detection (b).

5.4 UML Code Smell Detection

In this section, we will provide queries that enable finding smells in the UML class diagram.

Cyclic Dependency

Two or more units (e.g., classes, methods) mutually depend on each other.

```
1 MATCH (a)-[*]->(a)
2 RETURN a, path
```

Listing 5.33: Cypher query to identify UML cyclic dependency smell.

Message Chain

A client unit (e.g., method) calls another unit, which then in turn calls another unit, and so on (navigation through the class structure).

```
1 MATCH (a)-[r1]->(b)-[r2]->(c)-[r3]->(d)-[r4]->(e)
2 RETURN a,r1,b,r2,c,r3,d,r4,e
```

Listing 5.34: Cypher query to identify UML message chain smell.

Unutilized Abstraction

Not or barely used units (e.g., class or method).

```
1 MATCH (n)
2 WHERE n.isAbstract="true" and not (n)--()
3 RETURN n
```

Listing 5.35: Cypher query to identify UML unutilized abstraction smell.

Deep Hierarchy

An unnecessarily deep hierarchy.

```
MATCH (a)-[r1]->(b)-[r2]->(c)-[r3]->(d)
WHERE r1.Label ='generalization'
      and r2.Label ='generalization'
      and r3.Label ='generalization'
RETURN a,b,c,d,r1,r2,r3
```

Listing 5.36: Cypher query to identify UML deep hierarchy smell.

Multipath Hierarchy

A subtype inherits both directly and indirectly from a supertype.

```
MATCH (c)<-[r3]-(a)-[r1]->(b)-[r2]->(c)
WHERE r1.Label ='generalization'
      and r2.Label ='generalization'
      and r3.Label ='generalization'
RETURN a,b,c,r1,r2,r3
```

Listing 5.37: Cypher query to identify UML multipath hierarchy smell.

CHAPTER 6

Evaluation

This chapter will go through evaluation, including examples of different model instances developed in different modeling languages. Furthermore, we will discuss how a generic transformation framework benefits enterprise architects. Additionally, we will report the evaluation of smell detection on large sets of models. Lastly, we will report on the evaluation of State-of-the-Art tools provided by Gartner.

6.1 Generic Analysis Approach Evaluation

In this section, we will answer the first research question in this thesis.

6.1.1 Ecore Model Instance

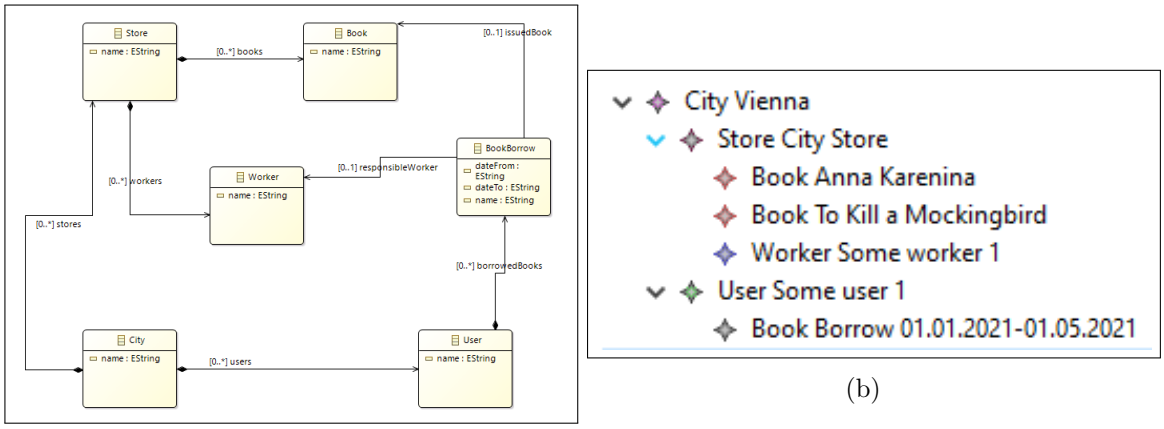
To show how it works with Ecore we have created a sample meta-model in Ecore (Fig. 6.1a) and a sample instance of the same meta-model (Fig. 6.1b).

The results of transformation of few tweaks of sample instance *ViennaBooks* can be seen in Fig. 6.2.

6.1.2 Archi Model Instance

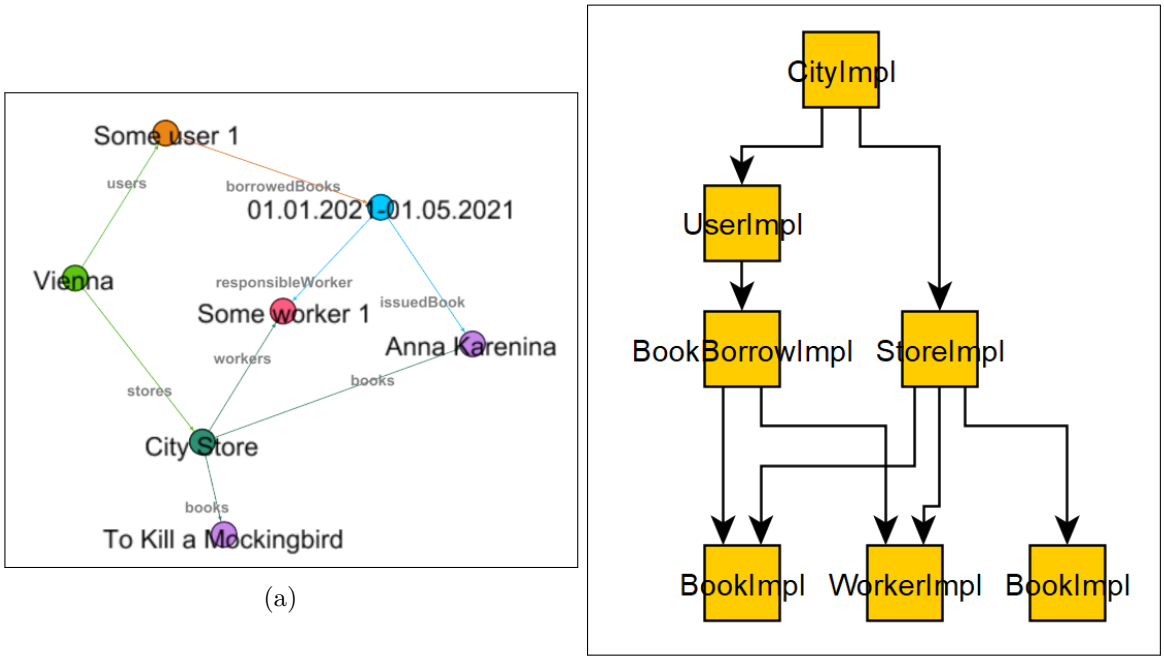
The example of transformation and analysis of the "App Store" Archi model will be shown. The results can be seen in Fig. 6.3 and Fig. 6.4.

As we can see, the model is successfully transformed with all the entities and relations. What is also more important is that in Fig. 6.3 we see the representation in Neo4j, and in Fig. 6.4 we see the model representation in another tools yED, and our platform CM2KG all look the same as it was the intention.



(a) (b)

Figure 6.1: Ecore sample meta-model *Vienna Books* specification (a), sample instance (b).

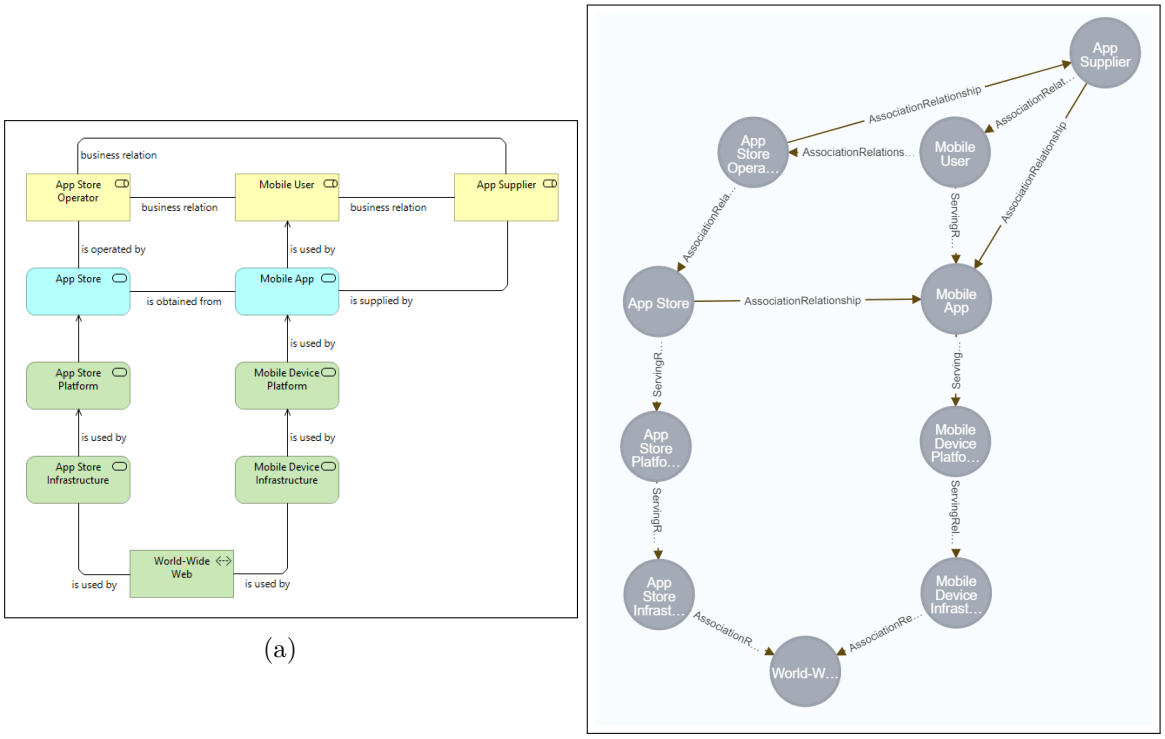


(a) (b)

Figure 6.2: Ecore sample meta-model *Vienna Books* resulting graph representation in Gephi (a), yEd (b).

6.1.3 ADOxx Model Instance

In this example, we will use the TEAM library. For the evaluation we created the same ArchiMetal Application Architecture model as in Fig. 6.9, and it can be seen in Fig, 6.5.



(a) (b)

Figure 6.3: Archi example *App store* intial model (a), representation in neo4j (b).

We use this example since it is an excellent example to show groups and nested graphs.

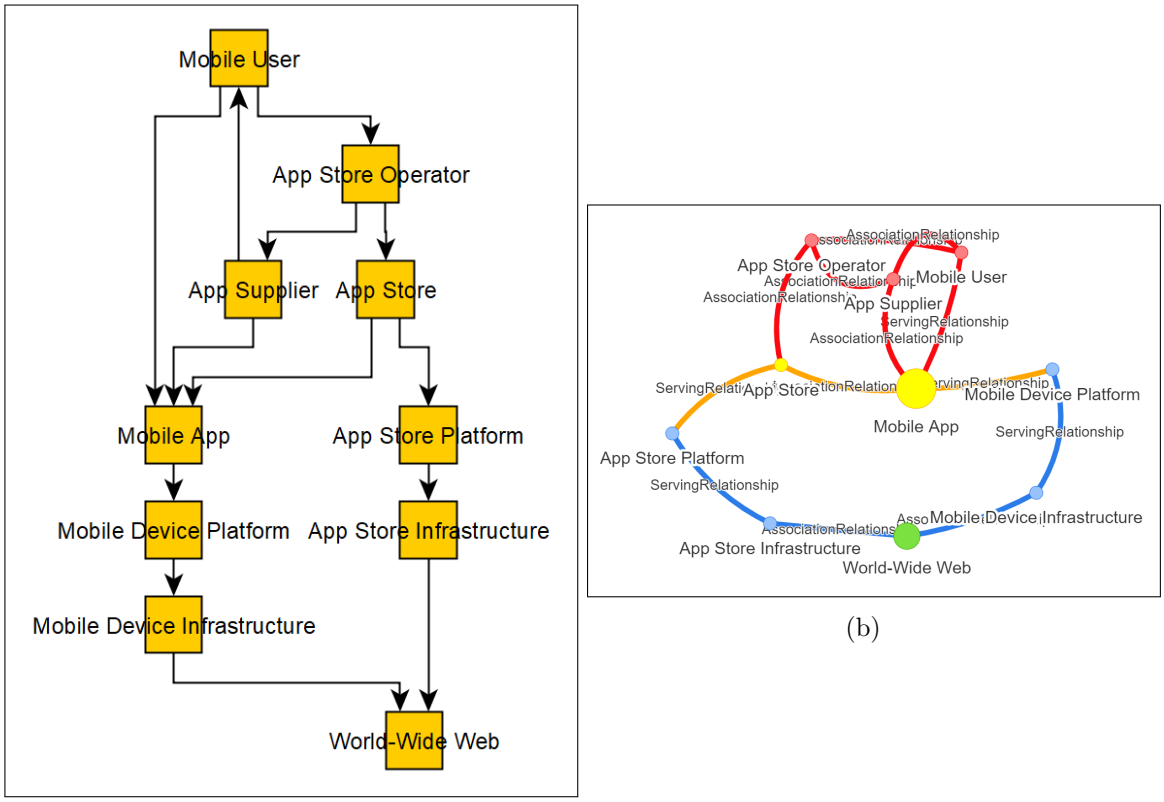
The result can be seen in Fig. 6.6 and in Fig. 6.7. An example of how the class properties are transformed is visible in the Fig.6.6a, where it can be seen how *Valid until* and *Notes* attributes are transformed. Furthermore, the grouping it can be best seen in Fig. 6.7b, where on the left side yEd shows nested graphs as folders which in our case were *Grouping* class instances.

6.1.4 Papyrus UML Model Instance

In this section, we will show an example of transformation from UML class diagram to GraphML. The result can be seen in Fig. 6.8. All the classes together with relations are successfully transformed. The result shown in Fig. 6.8 is the representation of the resulting graph in our platform CM2KG.

6.1.5 Generic Transformation Framework

To start with evaluating the general transformation framework, we will first define some requirements that will prove the generality. In order to state that this framework is generic, it should be able to fulfill the following requirements.



(a)

(b)

Figure 6.4: Archi example *App store* in yED (a), representation in CM2KG (b).

REQ1 A framework should be able to handle arbitrary conceptual modeling language.

REQ2 A framework should be able to handle models developed in different meta-modeling platforms.

REQ3 A framework should enable the usage of third-party graph analysis tools.

We start with the evaluation of REQ1. Here we need to check to what extent this approach enables usage of different modeling languages. For this, we used four different modeling languages: Ecore, Archimate, Papyrus UML, and TEAM [Bor+18]. These examples can be seen in Sec. 6.1.1, Sec. 6.1.2, Sec. 6.1.3, and Sec.6.1.4. Showing this on four popular different modeling languages checks REQ1 as fulfilled. The following requirement is related to different meta-modeling platforms. Similar to REQ1, we showed examples for models developed in four different modeling tools: Eclipse EMF, Archi, Eclipse Papyrus, and ADOxx. A similar claim as in the previous requirement follows here. We have shown examples of four different popular meta-modeling platforms, and we can state that REQ2 is also fulfilled. The third requirement is about enabling usage of different third-party graph analysis tools. To show generality here, we have selected three

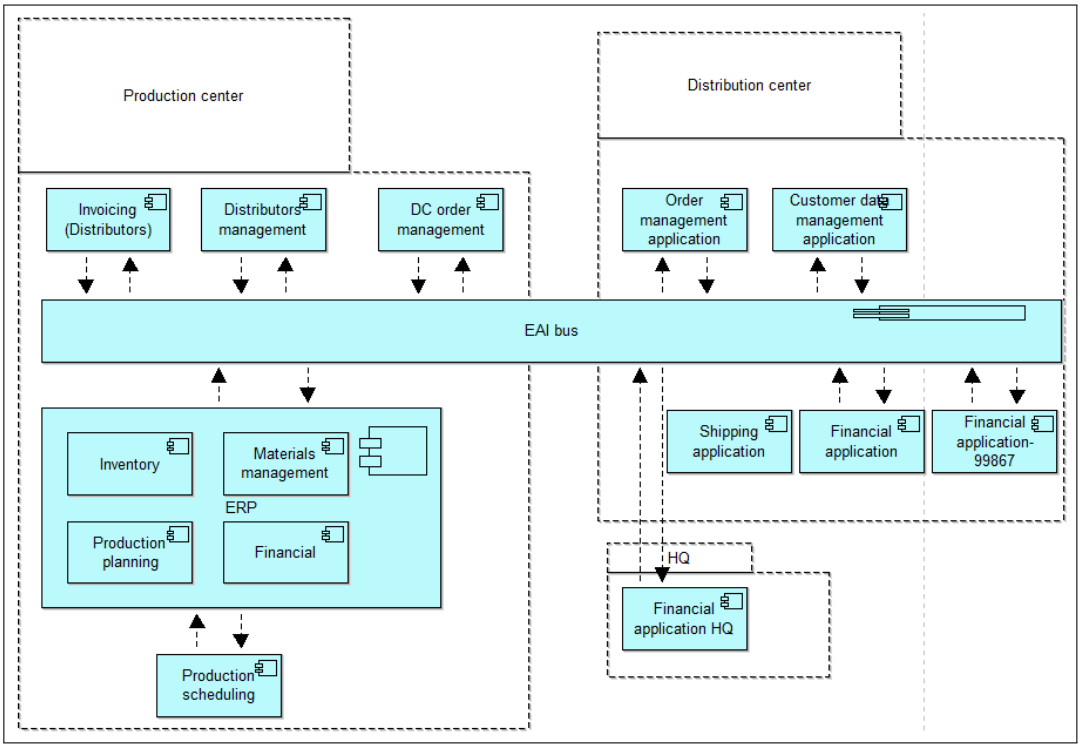
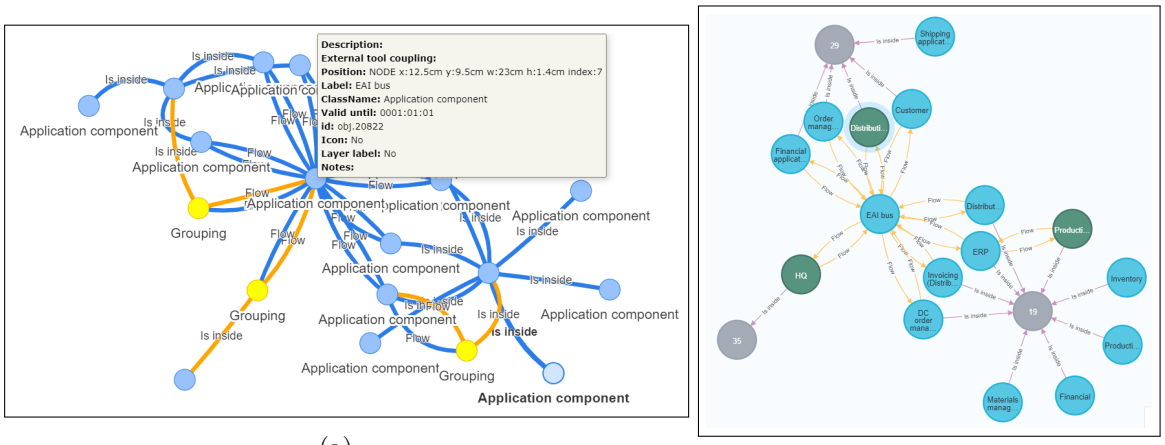


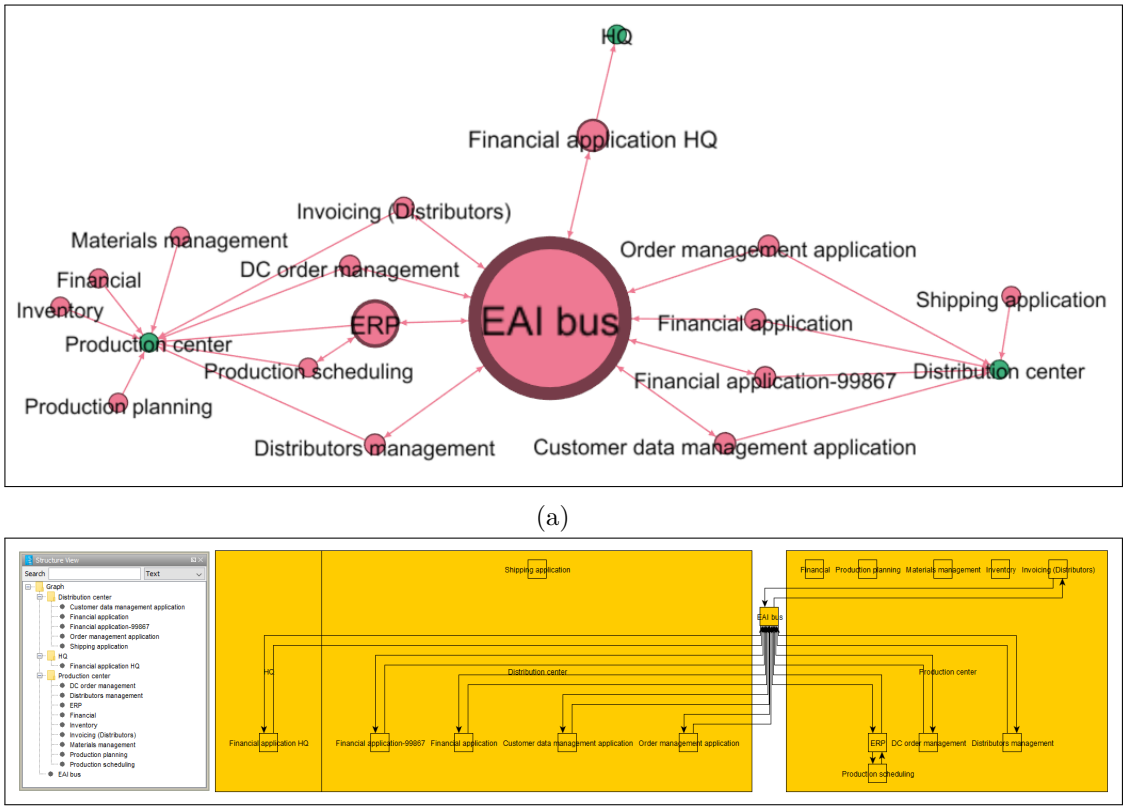
Figure 6.5: ADOxx exanple ArchiMetal-TEAM Application Architecture



(a) (b)

Figure 6.6: ADOxx exanple ArchiMetal-TEAM Application Architecture representation in CM2KG (a), in neo4j (b).

popular graph analysis tools. The first two are Gephi and yEd, by default graph analysis tools, and the third is Neo4j, a graph database that can be used for graph analysis. We will take one model and check it in these three different third-party tools.



(a)

(b)

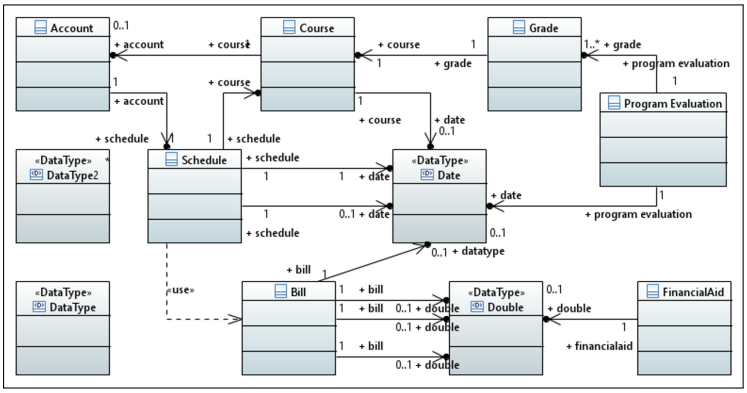
Figure 6.7: ADOxx exanple ArchiMetal-TEAM Application Architecture representation in Gephi (a), in yEd (b).

An example model we will use here to show generality using different third-part tools is the publicly available model ArchiMetal [Arc16], which can be seen in Fig. 6.9. How this model looks like in three different analysis tools can be seen in Fig. 6.10. The model shown in Fig. 6.9 is transformed to a graph using our platform and imported in these tools showing the same model with different features provided by tools. We can see that the transformed models look the same, and this sums up to a claim that REQ3 can also be checked as fulfilled.

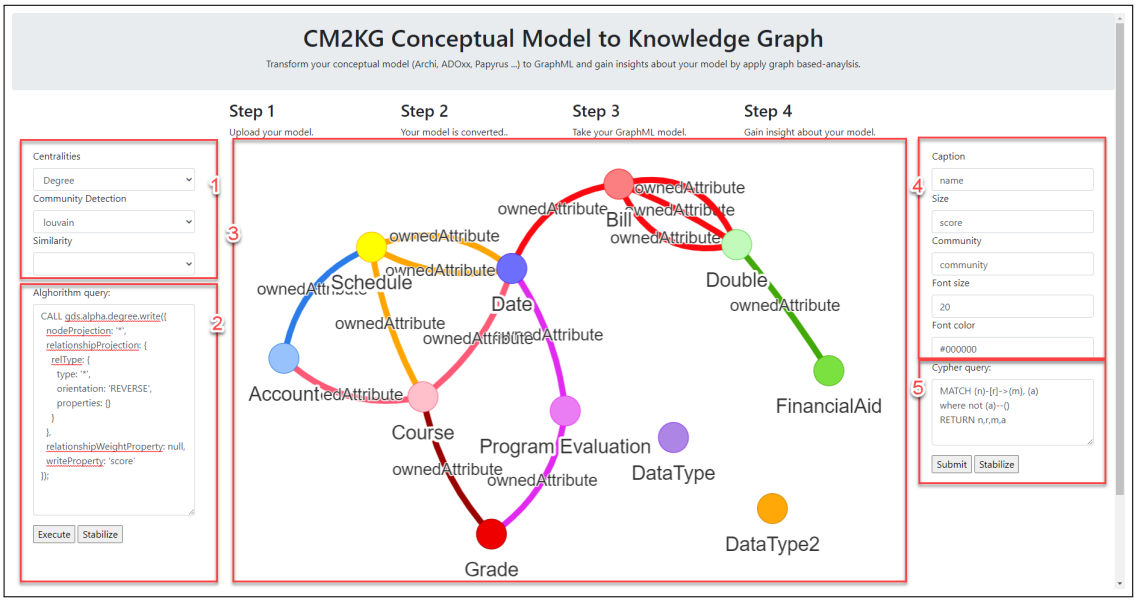
We can say that the framework is generic since we have shown that our prototype framework can use arbitrary modeling language, arbitrary meta-modeling platform, and arbitrary third-party graph analysis tool.

6.1.6 Benefits of Graph-Based Analysis for EAs

In the previous section, we showed that it is possible to accomplish the generic graph-based approach. Now we move further and discuss the benefits of using a graph-based



(a) The source UML model in Papyrus.



(b) The transformed Knowledge Graph in the CM2KG Cloud platform.

Figure 6.8: An example transformation from a UML model (a) to a Knowledge Graph (b) in the CM2KG Cloud platform. [SB21a]

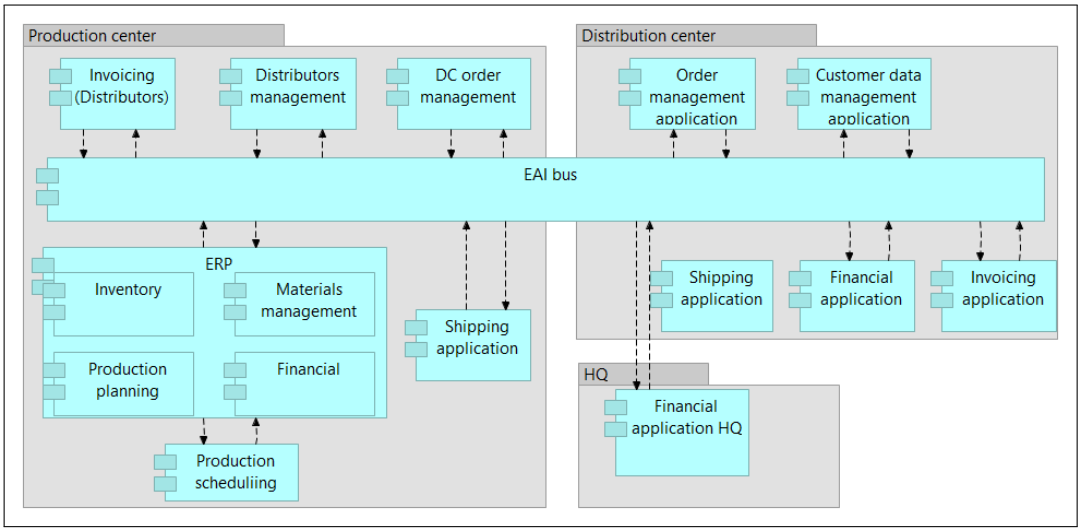


Figure 6.9: ArchiMetal Application Architecture [Arc16]

approach in analyzing conceptual models. In order to answer this, we refer to competency questions introduced in Sec. 5.2 and summarized in Tab. 5.2. We will use the same ArchiMetal [Arc16] model, and we will show examples of how graph centrality metrics and community detection metrics can be used to answer some competency questions.

We will start with applying betweenness, which is a graph centrality metric, to show how close the node is to others. The result of this we show in yEd tool, which can be seen in Fig. 6.10c. On the exact figure, it can be seen that *EAI bus* is much bigger and has a more intense red color. The reason for this is that in yEd we configured size of the node, and the intensity of the color depends on the betweenness value. This means that EAI bus is the node which is closest to all others which in reality totally makes sense since all data exchange would probably go through the enterprise service bus which in this case is called EAI bus.

The following example is related to the *PageRank* graph metric. We refer to the same model ArchiMetal [Arc16], but in this case, we set on a view called *Activity Model of Production Operations Management* which can be seen in Fig. 6.11 on the left side. On the right side, we see the values of PageRank graph centrality, which shows the importance of business functions to another. In PageRank, an entity with a higher score means more links are pointing to it, which is more authoritative. If we go back to our example, on the right side of Fig. 6.11 we see a listing of business functions scored based on PageRank value in descending order. We can interpret this listing so that *Production resource management* is more authoritative than *Production data collection*. In the real world, this can make sense if we see it in a way that management of production resources should come before production data collection.

The third example will be related to community detection. Again we used the same ArchiMetal [Arc16] model, and as an example, we applied *Strongly Connected Components*.

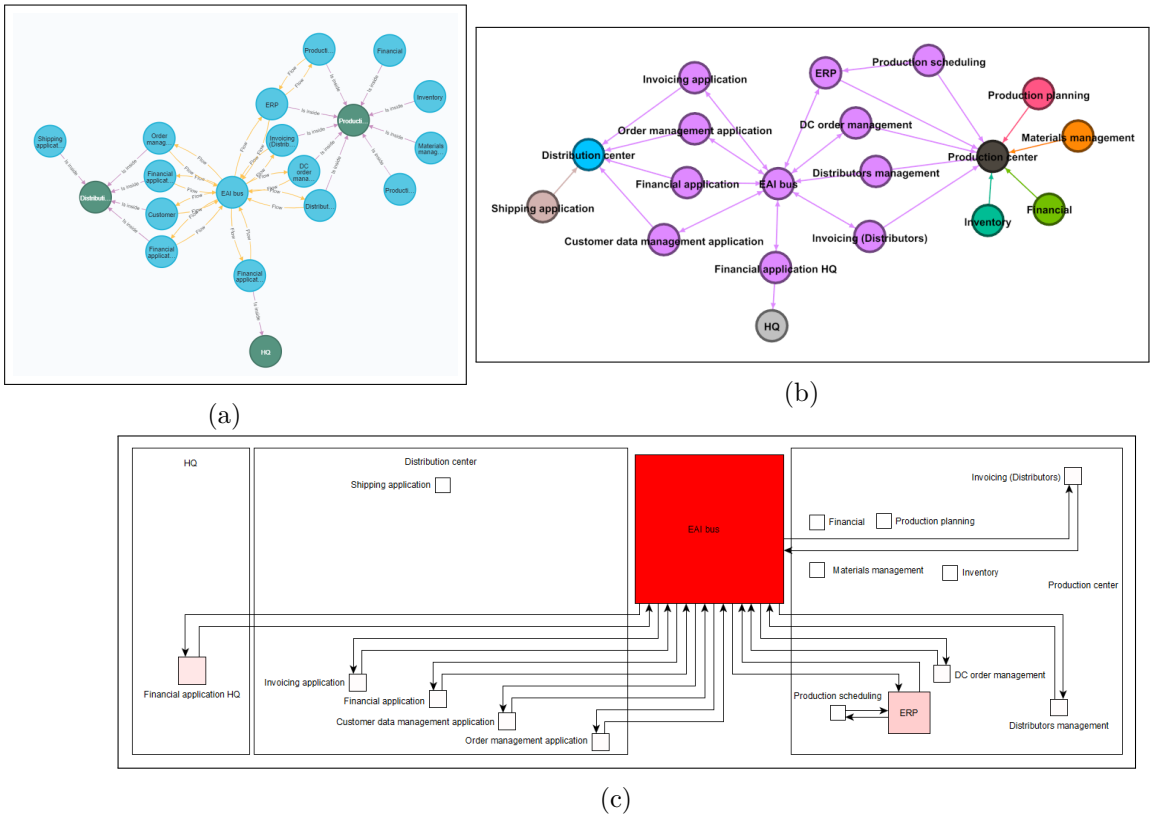
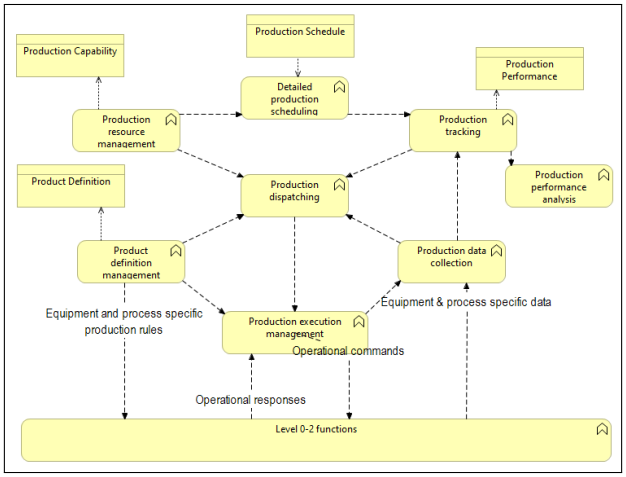


Figure 6.10: Transformed ArchiMetal example in Neo4j (a), Gephi (b), and yEd (c). [SB21b]

As a reminder, strongly connected components are the components where each node can access another node. This time we show the result in another tool, Gephi, and the result can be seen in Fig. 6.10b. Different communities are represented with different colors. If two nodes are the same color, they belong to the same community. Interestingly, this example is the community represented with purple color, which in the central part has *EAI bus*. If we go back to Fig. 6.9, we can easily see how the EAI bus has both: incoming and outgoing relationship to each entity, meaning information can go back and to. For the community we detected using strongly connected components, we can see how the algorithm identified this and knows that each application component in Fig. 6.9 can theoretically exchange information to another one (in graph-based words, *"they belong to the same strongly connected component"*).

More to add on graph-based analysis in the context of business perspective is that it can indicate which components can be a risk in running business operations in given application architecture. Examples we showed were in third-party graph analysis tools. However, the prototype platform CM2KG developed for the sake of this thesis can also provide the same answer for the example of betweenness which can be seen in Fig. 6.12.



PageRank	Business function
0.153904	Production resource mgmt
0.142801	Production definition mgmt
0.108743	Production execution mgmt
0.098338	Detailed production scheduling
0.096654	Production tracking
0.085166	Production data collection
0.061755	Production dispatching
0.030943	Product Definition

Figure 6.11: ArchiMetal Business Layer model and corresponding PageRank values.

On the exact figure, it can be seen how a user can select different graph centrality measure (refer to Tab. 5.2) and combine it with Cypher query to filter required result further.

We now discuss how a user can use the CM2KG platform in the given context. After the user has selected the corresponding query, it is presented in the textbox. The user can configure it further with different parameters, and the right side of the preview can apply different parameters for rendering the result. The last part is the Cypher query text box in the right corner which enables full possibilities for filtering based on user needs. Putting all together this example with betweenness (Fig. 6.12) demonstrates that CM2KG platform is capable of providing results as some popular tools. Of course, CM2KG has much fewer features, but it is a good example to demonstrate the generality of the framework and benefits of graph-based analysis for enterprise architects.

The previous three examples show a part of what can be done with the application of graph metrics, i.e., *Betweenness*, *Strongly Connected Components*, and *PageRank*, and how it can benefit enterprise architects. Obviously, much more can be done and is left for further research, but for the sake of evaluating RQ1, we have shown that there are analysis benefits by using a graph-based approach.

6.2 Analysis Automation Evaluation

This section will focus on evaluating appropriate means to automate graph-based analysis of EA models.

First, we define a set of requirements for automation of analyzing EA models. We set the following requirements in Tab. 6.1. Afterwards we will explore different ways in order to fulfill the requirements.

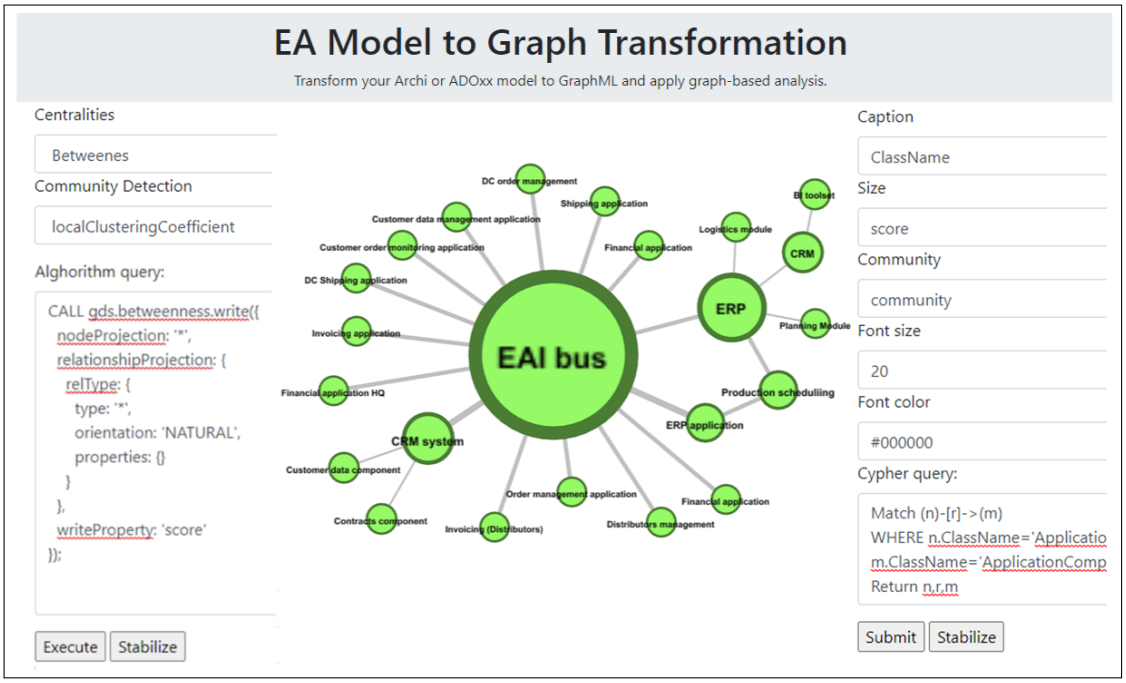


Figure 6.12: User Interface of the eGEAA platform. [SB21b]

No.	Requirement
REQ1	Automation should be supported by a tool.
REQ2	The tool should be running in the cloud.
REQ3	The tool should be able to accept EA models modelled with different languages.
REQ4	The tool should work with a widely supported file format for graphs.
REQ5	The tool should be able to transform the input model to a supported graph format.
REQ6	The tool should be able to export the transformed model.
REQ7	The tool should use well performing graph database engine.
REQ8	The tool should be able to execute queries on a graph database.
REQ9	The tool should visualise the results of a query.
REQ10	The tool should be extensible.

Table 6.1: List of requirements for automated analysis of EA models.

6.2.1 Analysis Tool Considerations

We can summarize REQ1 and REQ2 together. Nowadays, IT can highly achieve automation by using different software tools. The tool should be running in the cloud and should be easily accessible via browser to provide as much as possible to different users. This saves time and resources to the end-user. Combining this, it is clear that a cloud-based platform should achieve automation. There are many different options

on how implementors can realize a cloud-based solution, and this is evolving each day. Examples of back-end frameworks are Express, Django, Rails, Laravel, Spring, ASP.NET Core. On the other side, examples of front-end frameworks are: Angular, React, Vue, Ember, Backbone. Also important to note here are different application servers where the application should be running. Choosing the set of various options depends on the applicability and costs.

The third requirement is related to having a possibility to work with different EA frameworks. Many industry standards are being used, and automation should support analyzing them. There are frameworks like TOGAF, ArchiMate, BPMN, BIAN, Business Model Canvas, COBIT, IT4IT, ITIL, and many more.

Moving to the next requirement (REQ4), we state we need to select a file format for storing graph data. Here we also have many different options. These options differ in how the file is structured, i.e., XML-based or text-based. The next difference is how they store a data structure, i.e., adjacency list, edge list, adjacency matrix, incidence matrix. The next difference is in the level of details about the graph, i.e., details about the nodes or edges abstract types like sub-graph. There are following formats here: DGML, DotML, GEXF, GraphML, GXL, XGMML, DPT. GML. On the other hand, it is important to know what tools support the graph format and it is widely used.

Requirement five is related to a generic analysis approach. To achieve it, the tool should take different models and transform them into selected graph formats from the previous requirement. Then the tool could execute analysis in the same way as other models. There are many different formats on how the enterprise models are stored, and transforming these models into one or more graph-based structures is not easy.

Sixth requirement is to enable reusability. This means the tool should be able to export the graph-based format so it can be used in another existing graph-analysis tools and also provide users the features that other popular tools have. Example of these tools are Gephi, yED, Graphviz.

The seventh requirement is very important. So far, the tool should be running in the cloud and have the possibility to take different EA models and transform them to a graph structure. But now the question is how to make use of it. One option is to export the graph structure to be used with other tools. The second option is to perform the analysis in the tool directly. To do this efficiently and to enable different analysis possibilities with the graph, the tool should have a graph-based database in the background that will store the graph and operate on it. Different databases provide different performance scores, scaling possibilities, querying options, and algorithms. Options to consider here are Neo4j, ArangoDB, Dgraph, OrientDB, Amazon Neptune, DataStax, FlockDB, Casandra. Titan and many more. Another thing to consider here is the options for importing a graph. The selected database should support importing a graph format from the REQ4.

The eighth requirement is connected to a previous requirement. To provide the most features, a user should utilize the graph database in the background as much as possible. The tool should provide the functionality to execute the query directly on the database.

To provide this following should be considered. First, the graph database should have a connector for third-party tools to execute queries. Second, the database should return a query result so the tool can interpret it. This is dependent on the graph database itself, and an example is Neo4j which provides Bolt and HTTP protocols to connect. It gives library drivers for.NET, Java, JavaScript, Go, Python.

The following requirement (number nine) is related to the previous requirement. After the graph database returns, a query result tool should visualize this. As the second requirement states that the tool should be running in the cloud, this indicates that visualization is related to front-end technologies. This means different JavaScript libraries for visualizing graphs should be considered. These libraries are D3, Keylines, Vis.js, NeoVis.js, Sigma.js, Ogma, G6, Ngraph, React-force-graph.

The last requirement for the tool is that it should be extensible. As stated in the REQ3, tool should support different languages, and there are various analysis applications possible. This requirement is interconnected with the REQ2 since it depends on the selected technology. The tool should provide a set of core features (model transformation, graph querying, graph visualization) as default. Each specific application should be able to extend this to a particular need.

6.2.2 Prototype Tool Evaluation

In this section we evaluate the CM2KG prototype platform against the requirements stated in Tab. 6.1.

No.	Fulfilled	Note
REQ1	yes	CM2KG platform is a tool.
REQ2	yes	CM2KG is running in the cloud.
REQ3	partially	Platform accepts Open Group ArchiMate Model Exchange File Format, ADOxx XML files, Papyrus UML files.
REQ4	yes	CM2KG works with GraphML file format which is supported by various tools like Gephi, yED, neo4j.
REQ5	partially	CM2KG can transform ArchiMate Model Exchange Format, ADOxx XML and Papyrus XML formats to GraphML.
REQ6	yes	CM2KG provides *Download* link button to download the file and additionally can be shared via *URL* link.
REQ7	yes	CM2KG uses Neo4j graph database which is scored as one of the best graph databases by different sources.
REQ8	yes	CM2KG provides input field where user can enter a query and execute it against Neo4j.
REQ9	yes	CM2KG provides the graph visualisation result via neovis.js directly in the browser.
REQ10	partially	In order to add new new module in CM2KG it is necessary to develop new module in Spring Boot application.

Table 6.2: Evaluation of CM2KG against requirements in Tab. 6.1.

6.3 Smell Detection Evaluation

This section will evaluate how the graph-based approach handles smell detection in models. We will evaluate EA smell detection and UML code smell detection since they are very similar but also belong to totally different domains, which can further support the claim on the generality of the framework.

6.3.1 EA Model Smell Detection

First, we define a set of requirements that will lead evaluation in this section. We set the following requirements:

REQ1 The approach should be feasible to detect EA smells automatically in ArchiMate models.

REQ2 The approach should be similar in performance to other EA Smell detectors.

In addition to this, we want to find out about the quality of the models with respect to the number of smells.

To begin with, we need a repository where many models are available. We chose an openly available model repository in the MAR search engine [LC20]. This repository provided 369 ArchiMate models created with Archi. Dataset characteristics can be seen in Tab. 6.3. Models were collected and transformed via CM2KG platform, and EA smells defined in Sec. 5.3 were executed against these models in a custom-made experiment.

Table 6.3: Metadata of the EA Smells experiments

Metrics	**Value ± = standard deviation**
Models [total]	347
Avg. model size [Nodes]	51.41 ± 97.04
Avg. model size [Edges]	47.14 ± 70.23

In order to evaluate the first requirement, models collected in [LC20] repository had to be transformed to Open Group Exchange format first since CM2KG accepts this standardized format. As previously stated, we had a set of 369 ArchiMate models, and CM2KG successfully transformed 347, which is 94% of all models. The models that were not transformed had some encoding issues in entity naming or they were corrupted. Results show that 12 out of 13 EA smells we considered were found, as it can be seen in Fig. 6.13 (left). This shows that the first requirement can be fulfilled and that approach is feasible to detect EA smells automatically.

Moving to the following requirement, which is related to a performance, we can see the results in Fig. 6.14. Executing graph queries in knowledge graphs can be done efficiently

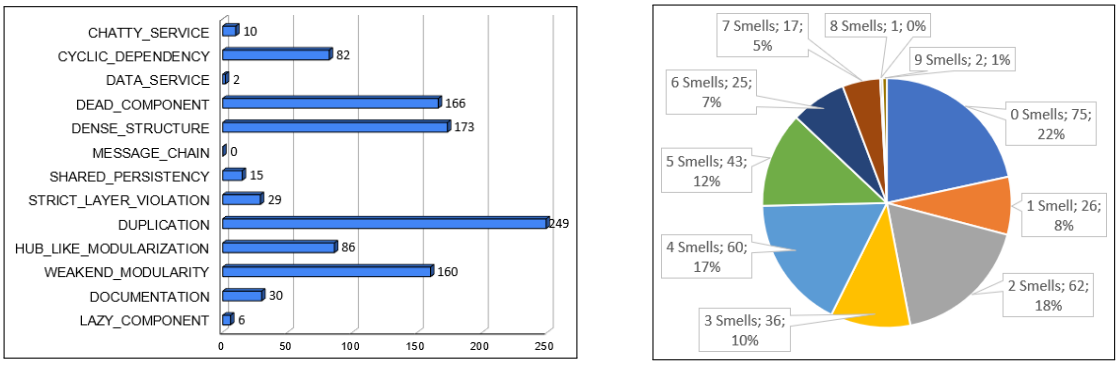


Figure 6.13: Detected EA Smells. [SHB21]

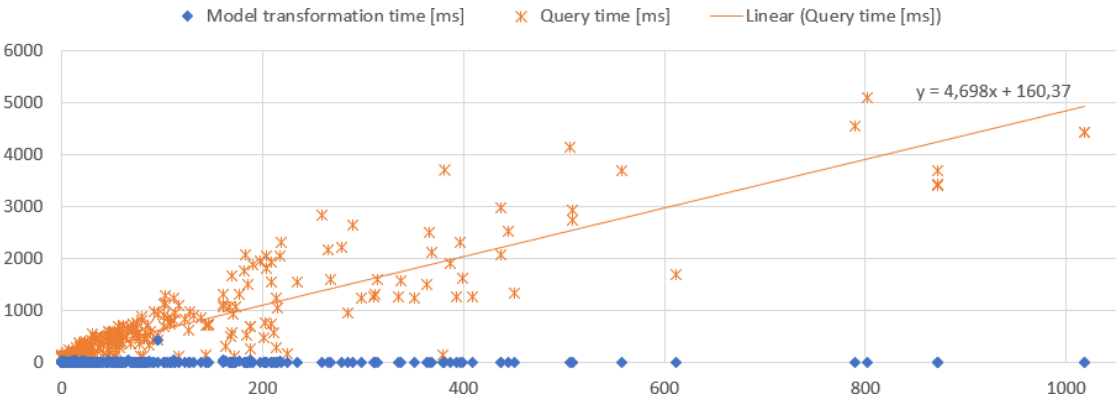
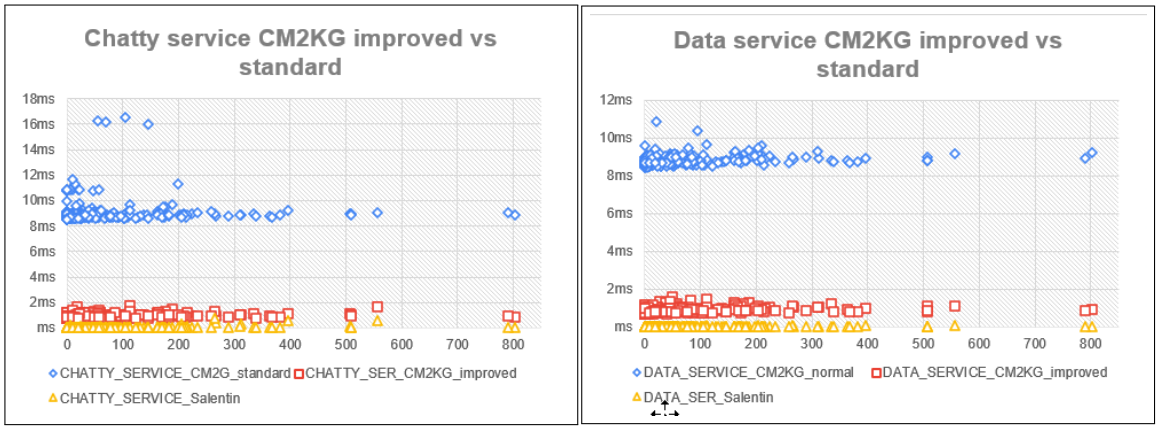


Figure 6.14: CM2KG performance of running all implemented EA Smells based on graph size. [SHB21]

on graphs with millions of nodes [Bel+19], and this can also be observed in our case of EA Smell detection which is basically executing a query on a graph. In Fig. 6.14 we can see the relation between the size of the model and the transformation time EA smell query execution time. Transformation time is shown in blue, and EA smell query execution is given in orange color.

We can observe the EA smell query execution performance has linear complexity according to the number of nodes. On the other hand, we can say the transformation has almost constant execution time. Larger models are unlikely to be seen in reality [Lag+13; Sch16]. We can also see that in our case, a model having more than 1000 elements (nodes and edges) was analyzed in a reasonable time. The average time for the model transformation is 4.46ms ±23.85ms, while the query execution average time is 623.41ms ±287.58ms depending on graph size.

Furthermore, to answer to the second requirement to have at least similar performance to other EA smell detectors, we compare the results of our experiment to the existing



(a) (b)

Figure 6.15: Salentin java implementation [SH20a] and CM2KG smell detection execution time comparison with initial vs improved queries for CM2KG.

Java-based implementation [SH20a] for EA smell detection. We have two types of results. First result represents the execution time measured in the CM2KG platform using the default queries and second one with improved query. The comparison (see Fig. 6.15) shows that on average, we have achieved 7.88ms faster time with improved query. We have to take into consideration here that Neo4j takes different actions in the background to execute a query which includes query planning, caching, indexing, etc. This means that with proper tweaking the execution time of CM2KG could be even better but this was out of scope of this thesis. Overall, results show that Java-based implementation outperforms our approach but only with a tiny difference. In the case of improved chatty service, this difference is only 0.89ms, and in the case of normal Neo4j query, Java-based implementation is 8.77ms faster on average. On the other hand, in some cases, the graph-based approach is even outperforming the Java-based implementation in case of cyclic dependency and message chain (see Fig. 6.16b and 6.16d).

Combining all these results, we can state that the second requirement is also fulfilled where it is important to note that our approach is generic while Java-based implementation is strictly made for one type of model. We did not invest time here to tweak the queries to execute them in the fastest possible way. Since neo4j offers tips for query execution improvement, one recommendation for future research could be to improve the performance results (one option would be to run the queries as stored procedures).

Last, we evaluate in this section the quality of the models from a given repository in the context of how many smells can be identified. The results can be seen in Fig. 6.13. The numbers show that only one specific smell exists in a model or not. We do not state how many times the same smell occurred in one exact model. An interesting result is that 78.38% had at least one smell. On the right side of Fig. 6.13 it can be seen that almost half of the models belong to a group of having no smell at all or having one smell or

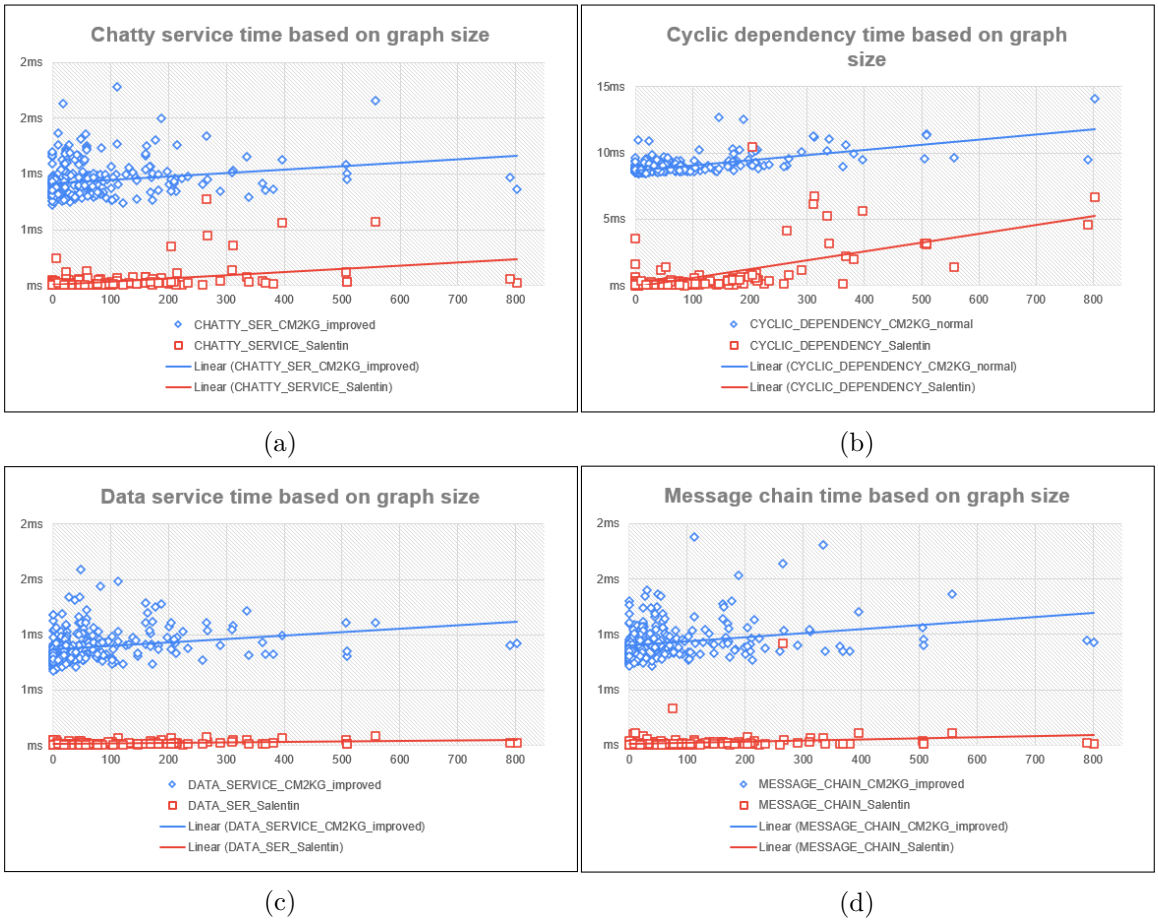


Figure 6.16: Salentin java implementation [SH20a] and CM2kg smell detection execution time comparison

having two smells.

Additional interesting question is what are the most frequent smells and what are the most frequent combinations of smells. The results showed that *Duplication* (249 hits), *Dense Structure* (173 hits), *Dead Component* (166 hits), and *Weakened Modularity* (160 hits) are the most frequent smells. The most frequent combinations of smells are *Dense Structure* and *Duplication* (162 hits), *Duplication* and *Weakened Modularity* (155 hits), and *Dead Component* and *Duplication* (153 hits). Combining the results here, we can say that the models that we were using here were quite "smelly".

6.3.2 Code smell

This section is similar to the previous section for the detection of EA smells. The detection of UML code diagram smells is very similar to what we already did in the previous section. The basis for EA smells was initially taken from UML code smells. We

have already introduced queries for the detection of UML smells in Sec. 5.4. The reason for evaluating this similar approach to EA smells is also to show the generality of the graph-based approach. We set the same requirements as for EA smells:

REQ1 The approach should be feasible to detect UML code smells automatically in UML class diagram models.

REQ2 The approach should be similar in performance to other code smell detectors.

In addition to this, we want to explore the quality of the models with respect to the number of smells.

We use the same repository [LC20] of models and we selected models created with the Papyrus [Gér+07] modeling tool. In total we have found 5.025 UML Class Diagram models in this repository created with Papyrus.

Again we set up a custom experiment to process all the models (i.e., transform it to GraphML and run the queries). Again, the platform could not transform all the models, resulting in a successful transformation of 4.262 (84.8%) out of 5.025 models. Although this is because some models contained special characters or were corrupted, this yields the limitation of the platform itself. Additional work has to be done to cover all the cases that can arise but overall 84.8% of the total size of 5.025 models can be considered as a success in order to automatically detect smells.

The results can be seen in Fig. 6.17. We were able to find all 5 smells that were described in Sec. 5.4. This yields a positive answer to the first requirement for this evaluation.

The second requirement to analyze the performance shows similar results as EA smells. The whole test (analysis of 5.025 models) lasted a bit more than 30 minutes. Average time to transform the model was 16ms (where the longest was 14 seconds for a model size larger than 100MB). The average time to execute a query was from 0.007s to 0.43s. We did not compare this directly to some other implementation, but we can state the performance is similar to the one in EA smells, and we can state that the second requirement is also fulfilled.

In the end, we report on the quality of the models with respect to 5 different UML code smell queries. Out of 4.262, we found 548 models that had at least one smell. Smell that had most hits (325 hits) was *Message Chain* followed by *Deep Hierarchy* (199 hits). With respect to having two different smells simultaneously in the model, we found 178 models. The most frequent combination of two different smells was *Message Chain* combined with *Deep Hierarchy* which we found in 102 models. The second most frequent combination was *Cyclic Dependency* with *Message Chain*, where we found 62 models with such smells.

6.3.3 Smell Detection Conclusion

Combining results from both EA smell detection and UML smell detection, we can see that in both cases, it is possible to set up the experiment using the graph-based approach

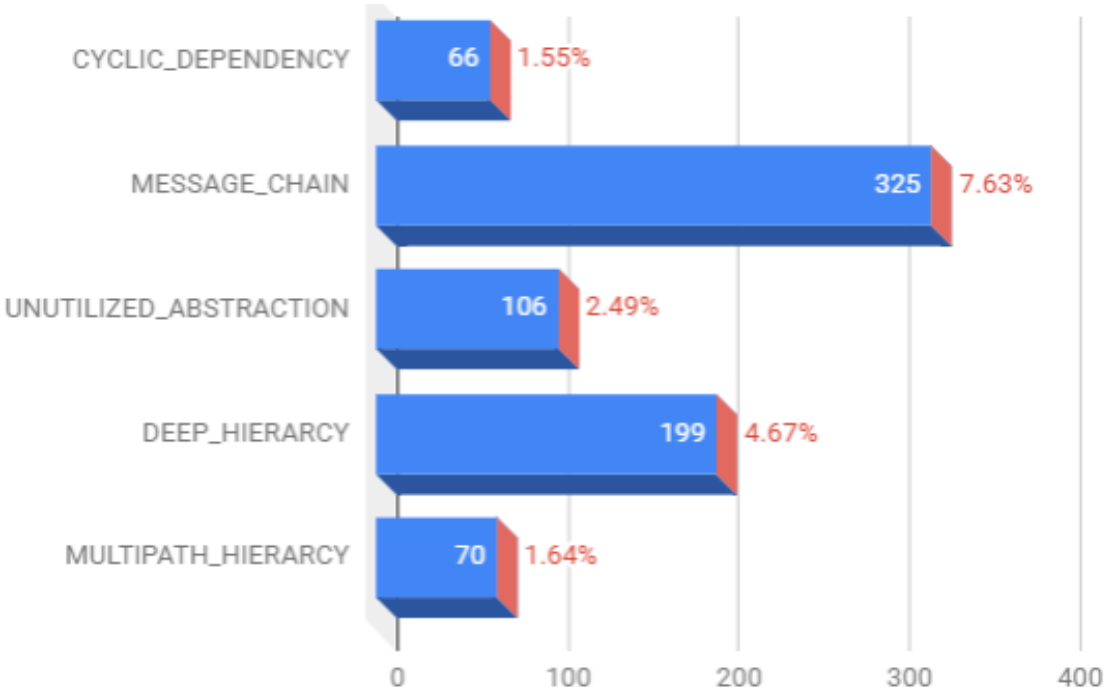


Figure 6.17: Code smells detected in 4.262 UML Class Diagram models. [SB21a]

to detect smells automatically. We also showed that many models in openly available model repositories contain many smells. Here we have to note that we do not guarantee the correctness of smell queries, but we showed how it can be used, and further research can deal with this since the point is not on the correctness of smell queries. Third and last, we showed that the approach performs head-to-head to current smell detection mechanisms and is scalable.

6.4 State of the Art Tools vs Graph-Based Analysis Comparison

The final evaluation result is given in the table based on the author's personal opinion supported by explanations for each tool. In the Tab. 6.4, we summarize features offered by different open source tools, some state-of-the-art tools and our prototype platform.

Commercial tools dominate the metric, sharing, collaboration features since they are built on ideology to provide valuable metrics and good business collaboration. Also, another dominant area of commercial tools is visual impact analysis since, again, this is important for businesses to see the actual results. On the other hand, open-source tools show that with simple plugins (*the idea of open-source software*), they can provide many features and be very useful.

Feature	Open Source			Commercial				
	Archi	Archi*	TEAM tool	ADOIT	ABACUS	BiZZDesign	Ardoq	CM2KG
Metrics overall	-	●	-	●	●	●	●	●
+Metrics calculation enabled	-	●	-	●	●	●	●	●
+Metrics calculation based on elements	-	●	-	●	●	●	●	●
+Metrics calculation based on relationships	-	●	-	●	●	●	●	●
Visual Impact Analysis overall	◐	●	◐	◐	●	●	●	●
+Predefined Views	●	●	-	●	●	●	●	-
+Customised views with filters	◐	●	-	◐	●	●	●	●
+Visualised impact view based on a specific query	◐	●	-	◐	◐	◐	●	●
+Visualised impact view based on a whole repository	◐	●	-	◐	●	●	●	●
Querying overall	-	●	◐	◐	◐	●	●	●
+Whole repository querying enabled	-	●	◐	◐	●	●	●	●
+Aggregated result for nodes and edges	-	●	-	-	◐	◐	●	●
+Specific user query creation	◐	●	◐	◐	◐	◐	●	●
Sharing and Collaboration overall	◐	-	◐	●	●	●	●	◐
+Online collaboration	-	-	◐	●	●	●	●	◐
+Online results sharing	-	-	-	●	●	●	●	◐
+Analysis export to pdf/image	●	●	●	●	●	●	●	◐
Technical debts overall	◐	●	-	◐	◐	◐	●	●
+EA Smell detection	◐	●	-	◐	◐	◐	●	●
+Predefined smells	◐	-	◐	◐	◐	●	◐	●
+Definition of new smells	-	●	-	-	-	-	●	●
Multiple models analysis	-	-	-	◐	◐	◐	◐	●
+Bulk file analysis report enabled	-	-	-	-	◐	◐	◐	●
+API for model analysis	-	-	-	●	●	●	●	◐

● = provides property; ◐ = partially provides property; - = does not provide property; *with Archimate Tool Database-Plugin

Table 6.4: Tools vs Graph-based Comparison with focus on analysis.

Sections, where platform CM2KG provides features are the querying section and multiple models analysis since it is mostly designed for that. The reasons for this are the options where the user can directly execute custom queries (Ardoq also supports this) and the openness of our platform where the platform is open source, and everybody can take the code and setup experiment in a way he needs. In the area of technical debts CM2KG provides features because it has many new smells implemented and can easily add new smells (again, Ardoq is not far from this since it can quickly implement the same smells). Overall we can say that, in this *artificial* setup of features survey, all three (open-source, state-of-the-art, prototype platform) provide a lot of features and can be very useful.

CHAPTER 7

Conclusion and Future Work

In this final chapter, we will summarize the findings of the thesis. First of all, we will analyze all the thesis results and validate all the defined research questions. Furthermore, we will mention the relevance of all contributions, and next, we will acknowledge the limitations. The final part will be about the foundations for future research in this and similar areas.

7.1 Summary

RQ1 The first research question is about accomplishing a generic approach for conceptual model analysis which is based on model transformation and graph structures. For this, we have developed a framework shown in Figure 4.1. In order to show to what extent this framework is generic, we used several different modeling languages: Ecore, ArchiMate, TEAM, and Papyrus UML. Furthermore, three different modeling platforms were used: Eclipse EMF, Archi, Eclipse Papyrus, ADOxx with TEAM library. On the other side, three independent third-party graph analysis tools were used: Neo4j, Gephi, yED. We have built a platform that enables the transformation of all these models into a graph-based structure that different graph analysis tools can use. This way, we have shown that we have covered some of the most common modeling languages and tools currently being used. Additionally, we have also shown that different graph analysis tools can be used. All of this adds to the generic nature of the framework. We have also provided examples of the benefits of how the graph-based analysis can be helpful and how it can answer some questions by enterprise architects by utilizing graph-based algorithms and query methods.

RQ2 The second question was about appropriate means to automate graph-based analysis of EA models. We have defined ten requirements based on the survey of state-of-the-art tools to answer this question. We have provided a set of possible

options to consider to fulfill each requirement. We have developed the CM2KG prototype platform[1], which tries to satisfy these requirements. We have analyzed the different options, and we have created a platform that accepts several different models modeled with varying modeling languages. The prototype platform is implemented as a Spring Boot application and is able to transform the Open Group ArchiMate Model Exchange File Format, ADOxx, and Papyrus UML file formats to a widely used GraphML format. The platform uses Neo4j graph database. In general, automation should be supported by a cloud-based tool that can accept different EA models. Moreover, the tool should be able to transform the input model to the graph format and store it in a well-performing graph database. The tool should provide a possibility to execute a query on a graph and visualize the results.

RQ3 The third question was about whether the graph-based analysis approach can detect EA model smells and UML model smells. In order to answer this question, we used the generic framework discussed in the first question, which was built into our platform. We have developed queries to identify 13 EA smells and five queries for identifying UML code smells. After some modifications to enable large-scale experiments, we examined 369 ArchiMate models and found out that more than 78% models had at least one smell. On the other side, we found a more extensive model repository containing 5.025 UML class diagram models for UML code smells. From 4.262 successfully processed models in the platform, a bit less than 13% of models had at least one smell. We have shown that the approach can automatically detect smells in both cases. Performance-wise, this general approach is not falling behind the detectors written in Java, and for models with a large number of nodes, it can even outperform current implementations by fully utilizing graph database engines.

RQ4 The fourth question was how the graph-based analysis approach and platform stand against the State-of-the-Art EA tools. In order to cope with this question, we first had to define the State-of-the-Art tools. For this, we have referred to the well-known Gartner magic quadrant. Each year, Gartner publishes the report on best Enterprise Architecture Tools calculated on many different criteria that were not the scope of this work, so we took the Leaders from this quadrant. Since all these tools are commercial and not all offer free trials, we took 4/8 tools and analyzed them with open source alternatives Archi and ADOxx. In total, we compared six commercial and open source with the prototype platform we developed for this thesis. Most commercial tools share the same preferences to provide well-defined and well-structured reporting, integrated collaboration, and integrations to many third-party platforms. Platforms also provide a possibility to work with many different modeling languages. All commercial platforms are running in the cloud and are accessible via browser. On the other hand, open-source tools we analyzed Archi and ADOxx are running on a local machine where Archi collaboration is

[1]CM2KG platform [online]: `https://me.big.tuwien.ac.at/`

not possible by default(although the plugin for collaboration can be installed), whereas ADOxx uses MS SQL database in the background and supports multiple users. All the commercial and open-source tools had the possibility to create and modify the models, whereas our prototype platform had only an import model as a file option. Interestingly, 2 out of 4 commercial tools (defined as Leaders) we analyzed are running graph databases in the background (Avolution ABACUS and Ardoq). Avolution ABACUS team even claims that they have used graph database technology for over 15 years because relational databases are a source of frustration because they cannot fulfill increasing user demands and analysis techniques. On the other hand, Ardoq even offers a user interface where, apart from all features offered by default, users can write and execute their custom Gremlin queries on the database containing model repositories in the background to get relevant information. We have summarized some high-level analysis features offered by tools that represent commercial leaders, open-source tools, and our prototype platform in one table. Coming back to the question of how graph-based analysis compares to analysis techniques provided by State-of-the-Art, we can say that some leaders in this area are already utilizing the graph-based approach and claim it opens new possibilities. Therefore we can say that the graph-based analysis approach of Enterprise Architecture has taken place in State-of-the-Art.

7.2 Contributions

This work has many artifacts as contributions to the community. First of all, a generic framework can be used as a guideline on how a conceptual model can be transformed into a graph structure and then analyzed by utilizing graph methods. In order to show the generic nature of the framework, we have developed four different transformations: Ecore2Graphml, ADOxx2Graphml, PapyrusUML2Graphml, Archimate2Graphml. We have created a set of rules, pseudo code, and actual Java implementation code for each transformation. Furthermore, we have built a whole web platform that should serve as an analysis platform for conceptual models based on a graph-based approach. This platform is intended to be expanded furthermore with new modules. The platform supports four different input models and can transform the models into graph structure file and also load it into the background neo4j database and visualize a result directly in the web browser. The platform also offers an API to upload and transform a model. Next, a list of possible metrics for analyzing enterprise architecture was proposed based on the goal question metric approach. This list was then summarized in a table of exemplary competency questions for Enterprise Architects and the corresponding graph metrics. An example of how a graph-based approach can answer each of these questions is provided. Additional contribution is the development of 13 different queries for detecting EA smells and five different queries for detecting UML code smells. Each smell is provided with an example. Last but not least, a comparison of different State-of-the-Art EA tools is summarized and evaluated against a graph-based analysis approach.

7.3 Limitations

Building up the transformation was one of the most challenging tasks since many models contained special characters (like in Archi) that were not acceptable in graph tools like Gephi or neo4j. All these special characters had to be handled manually. As it was tough to get professional feedback from the people working in the EA area, we do not know what the opinion of the real stakeholders and practitioners about our approach and our platform is. Another issue is the lack of real-world models that can be used for testing. Repositories that we used to take the models are primarily models for academic and test purposes, so we lacked real models from the industry. This made it harder to find some good examples of where a graph-based approach can be utilized.

7.4 Future Research

The following are the recommendations for future research. As not much is spent on evaluating the correctness of the transformations, this can be evaluated more in detail. The platform we developed is intended to be extended with new modules for analysis. The platform lacks integration options, and features that would be beneficial are, for example:

- REST interface
- Archi plug-in

At the moment of writing this thesis, there are already other theses that have started to expand the platform. On the other side, many options are available on applying the graph methods and possibilities for analyzing EAs. Another open topic is the correctness of implemented smells and the development of new ones. More can also be done on evaluating different background graph database engines where maybe performance can be improved. Another unexplored area is how this approach can be combined with semantic web standards RDF/OWL. As we were limited to checking with the real practitioners on the usability, more research can be done here.

List of Figures

List of Tables

List of Algorithms

Acronyms

ADOxx ADOxx. 4–6, 15, 16, 39, 40, 47, 50–52, 59, 62, 115, 117

Archi Archi. 4, 5, 39, 54, 55, 62

ArchiMate ArchiMate. 2–5, 13, 14, 28, 40, 41, 54, 55, 59, 62

CM2KG Conceptual Model to Knowledge Graph Platform. 57–59, 61, 63–65, 67, 68, 87, 89–91, 93, 95, 96, 99–102, 106, 108, 112, 113, 115

DS Design Science. 5

DSRM Design Science Research Methodology. 5

EA Enterprise Architecture. 1–4, 27, 28, 31, 46, 47, 61, 96–98, 107, 108, 110, 115

EA Modeling Enterprise Architecture Modeling. 1, 2, 14

EAM Enterprise Architecture Management. 9, 10, 14, 35, 111

Ecore Ecore. 4–6, 13, 47–51, 54, 55, 62, 87, 88, 112

Gephi Gephi. 6, 48, 64

GraphML GraphML. 6, 19–21, 47–55, 58, 61–64, 89, 99, 104, 108, 115, 117

Neo4j Neo4j. 6, 48, 63–65, 87, 99, 102, 108

TEAM TOGAF based Enterprise Architecture Management. 5

TOGAF The Open Group Architecture Framework. 10

XML Extensible Markup Language. 51, 61, 63

yED yED. 4, 6, 48, 64, 87

Bibliography

[ACB14] Concalo Antunes, Artur Caetano, and José Borbinha. "Enterprise Architecture Model Analysis Using Description Logics". In: Sept. 2014, pp. 237–244. DOI: 10.1109/EDOCW.2014.43.

[ADO21a] ADOxx. *ADOxx Home*. [Online; accessed 23.12.2021] https://www.adoxx.org/. 2021. URL: https://www.adoxx.org/. (accessed: 23.12.2021).

[ADO21b] ADOxx. *Introduction to ADOxx*. [Online; accessed 23.12.2021] https://www.adoxx.org/live/introduction-to-adoxx. 2021. URL: https://www.adoxx.org/live/introduction-to-adoxx. (accessed: 23.12.2021).

[AG] Software AG. *Software AG Alfabet*. [Online; accessed 28.11.2021] https://www.softwareag.com/en_corporate/platform/alfabet/enterprise-architecture.html. URL: https://www.softwareag.com/en_corporate/platform/alfabet/enterprise-architecture.html. (accessed: 28.11.2021).

[Ahl+12] Frederik Ahlemann et al. *Strategic Enterprise Architecture Management*. Berlin, Heidelberg: Springer, 2012.

[Aie06] Stephan Aier. "How Clustering Enterprise Architectures helps to Design Service Oriented Architectures". In: *2006 IEEE International Conference on Services Computing (SCC 2006), 18-22 September 2006, Chicago, Illinois, USA*. IEEE Computer Society, 2006, pp. 269–272. DOI: 10.1109/SCC.2006.52. URL: https://doi.org/10.1109/SCC.2006.52.

[Ant+14] Gonçalo Antunes et al. "Ontology-based enterprise architecture model analysis". In: (Mar. 2014). DOI: 10.1145/2554850.2555176.

[Arc16] Archi. *ArchiMetal*. https://github.com/archimatetool/ArchiModels/tree/master/ArchiMetal. 2016.

[Arda] Ardoq. *Ardoq Enterprise Architecture*. [Online; accessed 28.11.2021] https://www.ardoq.com/. URL: https://www.ardoq.com/. (accessed: 28.11.2021).

[Ardb] Ardoq. *Ardoq Live Product Demo with Ian Stendera*. [Online; accessed 30.11.2021] https://www.youtube.com/watch?v=8I5NTUue1h8&list=PLN64rGvDdWmlc3FDmT_2BdcV23HNJskcn&index=5. URL: https://www.youtube.com/watch?v=8I5NTUue1h8&list=PLN64rGvDdWmlc3FDmT_2BdcV23HNJskcn&index=5. (accessed: 30.11.2021).

[Ardc] Ardoq. *How to do as-is and to-be analysis in Ardoq*. [Online; accessed 30.11.2021] https://www.youtube.com/watch?v=-8hKta9OaMs&list=UU668g33FO7vdGI7pwZt4ruw&index=31. URL: https://www.youtube.com/watch?v=-8hKta9OaMs&list=UU668g33FO7vdGI7pwZt4ruw&index=31. (accessed: 30.11.2021).

[AT10] Thorsten Arendt and Gabriele Taentzer. *UML model smells and model refactorings in early software development phases*. Tech. rep. Universitat Marburg, 2010.

[Avoa] Avolution. *Avolution Abacus*. [Online; accessed 28.11.2021] https://www.avolutionsoftware.com/enterprise-architecture/. URL: https://www.avolutionsoftware.com/enterprise-architecture/. (accessed: 28.11.2021).

[Avob] Avolution. *The Graph Database Advantage for Enterprise Architects*. [Online; accessed 04.12.2021] https://www.avolutionsoftware.com/abacus/the-graph-database-advantage-for-enterprise-architects/. URL: https://www.avolutionsoftware.com/abacus/the-graph-database-advantage-for-enterprise-architects/. (accessed: 04.12.2021).

[Bar+19] Amanda Oliveira Barbosa et al. "A Taxonomy for Enterprise Architecture Analysis Research". In: *Proceedings of the 21st International Conference on Enterprise Information Systems, ICEIS 2019, Heraklion, Crete, Greece, May 3-5, 2019, Volume 2*. Ed. by Joaquim Filipe et al. SciTePress, 2019, pp. 493–504. DOI: 10.5220/0007692304930504. URL: https://doi.org/10.5220/0007692304930504.

[Bel+19] Luigi Bellomarini et al. "Knowledge graphs and enterprise AI: the promise of an enabling technology". In: *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE. 2019, pp. 26–37.

[BHM20] Jan vom Brocke, Alan Hevner, and Alexander Maedche. "Introduction to Design Science Research". In: Sept. 2020, pp. 1–13. ISBN: 978-3-030-46780-7. DOI: 10.1007/978-3-030-46781-4_1.

[BiZa] BiZZDesign. *HoriZZon Demo 2.* [Online; accessed 04.12.2021] https://www.youtube.com/watch?v=JmLHOPZ5YwQ&t=27s. URL: https://www.youtube.com/watch?v=JmLHOPZ5YwQ&t=27s. (accessed: 04.12.2021).

[BiZb] BiZZDesign. *HoriZZon Demo 5*. [Online; accessed 04.12.2021] https://www.youtube.com/watch?v=Md4A6IW08TY&t=527s. URL: https://www.youtube.com/watch?v=Md4A6IW08TY&t=527s. (accessed: 04.12.2021).

[BiZc] BiZZdesign. *BiZZdesign Enterprise Architecture*. [Online; accessed 28.11.2021] https://bizzdesign.com/solution/enterprise-architecture/. URL: https://bizzdesign.com/solution/enterprise-architecture/. (accessed: 28.11.2021).

[BJS13] Markus Buschle, Pontus Johnson, and Khurram Shahzad. "The Enterprise Architecture Analysis Tool - Support for the Predictive, Probabilistic Architecture Modeling Framework". In: *19th Americas Conference on Information Systems, AMCIS 2013, Chicago, Illinois, USA, August 15-17, 2013*. Association for Information Systems, 2013. URL: http://aisel.aisnet.org/amcis2013/EnterpriseSystems/GeneralPresentations/15.

[BJT16] Francis Bloch, Matthew Jackson, and Pietro Tebaldi. "Centrality Measures in Networks". In: *SSRN Electronic Journal* (Aug. 2016). DOI: 10.2139/ssrn.2749124.

[BMS09] Sabine Buckl, Florian Matthes, and Christian M Schweda. "Classifying enterprise architecture analysis approaches". In: *IFIP-International Workshop on Enterprise Interoperability*. Springer. 2009, pp. 66–79.

[Bor+18] Dominik Bork et al. "Requirements Engineering for Model-Based Enterprise Architecture Management with ArchiMate". In: *Proceedings EOMAS 2018*. Springer, 2018, pp. 16–30.

[BP05] Ulrik Brandes and Christian Pich. "GraphML Transformation". In: *Graph Drawing*. Ed. by János Pach. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 89–99. ISBN: 978-3-540-31843-9.

[Bra+02] Ulrik Brandes et al. "GraphML Progress Report Structural Layer Proposal". In: *Graph Drawing*. Ed. by Petra Mutzel, Michael Jünger, and Sebastian Leipert. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 501–512. ISBN: 978-3-540-45848-7.

[Bra+13] Ulrik Brandes et al. "Graph Markup Language (GraphML)". In: *Handbook of graph drawing visualization*. Ed. by Roberto Tamassia. Discrete mathematics and its applications. CRC Press, 2013, pp. 517–541.

[BS21] Dominik Bork and Muhamed Smajevic. *Companion source code repository of the eGEAA platform*. https://github.com/borkdominik/eGEAA. 2021.

[BW84] Victor R. Basili and David M. Weiss. "A Methodology for Collecting Valid Software Engineering Data". In: *IEEE Transactions on Software Engineering* SE-10.6 (1984), pp. 728–738. DOI: 10.1109/TSE.1984.5010301.

[Cap] Capsifi. *Capsifi Enterprise Architecture*. [Online; accessed 28.11.2021] https://www.capsifi.com/solutions/architecture/. URL: https://www.capsifi.com/solutions/architecture/. (accessed: 28.11.2021).

[Con21] Neo4j Contrib. *neovis.js*. https://github.com/neo4j-contrib/neovis.js. https://github.com/neo4j-contrib/neovis.js. 2021.

[CSQ08] Weiwei Cui, Supervisor, and H. Qu. "A Survey on Graph Visualization". In: 2008.

[Cun92] Ward Cunningham. "The WyCash Portfolio Management System". In: *SIGPLAN OOPS Mess.* 4.2 (Dec. 1992), pp. 29–30. ISSN: 1055-6400.

[DA09] M. R. Davoudi and F. S. Aliee. "Characterization of Enterprise Architecture quality attributes". In: *2009 13th Enterprise Distributed Object Computing Conference Workshops*. 2009, pp. 131–137. DOI: 10.1109/EDOCW.2009.5332004.

[Deh+14] Matthias Dehmer et al. "What Is Quantitative Graph Theory?" In: Nov. 2014, pp. 1–33.

[DES17] Matthias Dehmer, Frank Emmert-Streib, and Yongtang Shi. "Quantitative Graph Theory: A new branch of graph theory and network science". In: *Information Sciences* 418-419 (2017), pp. 575–580.

[DI06] D. Dreyfus and B. Iyer. "Enterprise Architecture: A Social Network Perspective". In: *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06)*. Vol. 8. 2006, 178a–178a. DOI: 10.1109/HICSS.2006.155.

[Ecl21a] Eclipse. *Eclipse Modeling Framework (EMF)*. [Online; accessed 23.12.2021] https://www.eclipse.org/modeling/emf/. 2021. URL: https://www.eclipse.org/modeling/emf/. (accessed: 23.12.2021).

[Ecl21b] Eclipse. *Package org.eclipse.emf.ecore*. [Online; accessed 05.01.2022] https://download.eclipse.org/modeling/emf/emf/javadoc/2.9.0/org/eclipse/emf/ecore/package-summary.html. 2021. URL: https://download.eclipse.org/modeling/emf/emf/javadoc/2.9.0/org/eclipse/emf/ecore/package-summary.html. (accessed: 05.01.2022).

[ecl21] eclipse.org. *Ecore*. [Online; accessed April 10, 2021], https://www.eclipse.org/Xtext/documentation/308_emf_integration.html. 2021. URL: https://www.eclipse.org/Xtext/documentation/308_emf_integration.html.

[EK13] Wendy Ellens and Robert Kooij. "Graph measures and network robustness". In: (Nov. 2013).

[FK15] Hans-Georg Fill and Dimitris Karagiannis. “On the Conceptualisation of Modelling Methods Using the ADOxx Meta Modelling Platform”. In: (Jan. 2015). DOI: 10.18417/emisa.8.1.1.

[Fow99] Martin Fowler. *Refactoring - Improving the Design of Existing Code*. Addison Wesley object technology series. Addison-Wesley, 1999.

[Gara] Gartner. *Company official website*. [Online; accessed 28.11.2021] https://www.gartner.com/en. URL: https://www.gartner.com/en. (accessed: 28.11.2021).

[Garb] Gartner. *Magic Quadrant for Enterprise Architecture 2021*. [Online; accessed 28.11.2021] https://www.softwareag.com/en_corporate/platform/alfabet/ea-tools-gartner.html. URL: https://www.softwareag.com/en_corporate/platform/alfabet/ea-tools-gartner.html. (accessed: 28.11.2021).

[Gér+07] Sébastien Gérard et al. “Papyrus: A UML2 tool for domain-specific language modeling”. In: *Dagstuhl Workshop on Model-Based Engineering of Embedded Real-Time Systems*. Springer. 2007, pp. 361–368.

[GKC06] Aditya Garg, Rick Kazman, and Hong-Mei Chen. “Interface descriptions for enterprise architecture”. In: *Science of Computer Programming* 61.1 (2006), pp. 4–15.

[Gro] BOC Group. *ADOIT Enterprise Architecture Suite*. [Online; 28.11.2021] https://www.boc-group.com/en/adoit/. URL: https://www.boc-group.com/en/adoit/. (accessed: 28.11.2021).

[Gro12] The Open Group. *Archimate 2.0 Specification*. 2012.

[Gro18] The Open Group. *The TOGAF® Standard, Version 9.2*. https://www.opengroup.org/togaf. 2018.

[Gro21] The Open Group. *ArchiMate® 3.1 Specification, a Standard of The Open Group*. [Online; accessed April 10, 2021] https://pubs.opengroup.org/architecture/archimate3-doc/toc.html. 2021. URL: https://pubs.opengroup.org/architecture/archimate3-doc/toc.html.

[Hac+19] Simon Hacks et al. “Towards the Definition of Enterprise Architecture Debts”. In: *IEEE 23rd International Enterprise Distributed Object Computing Workshop (EDOCW)*. IEEE, Oct. 30, 2019, pp. 9–16. ISBN: 978-1-7281-4598-3. DOI: 10.1109/EDOCW.2019.00016.

[Hae18] Thorsten Haendler. “On using UML Diagrams to Identify and Assess Software Design Smells”. In: *Proceedings of the 13th International Conference on Software Technologies*. SciTePress, 2018, pp. 447–455.

[HC10] Alan Hevner and Chatterjee. *Design Research in Information Systems: Theory and Practice*. Vol. 22. Jan. 2010. DOI: 10.1007/978-1-4419-5653-8.

[Hew19] Richard Heward. *Enterprise Architecture Analysis in a Graph Database*. [Online; accessed 16.01.2022] https://www.tamebluelion.co.uk/EA-in-a-graph. 2019. URL: https://www.tamebluelion.co.uk/EA-in-a-graph. (accessed: 16.01.2022).

[Hol+09] Oliver Holschke et al. "Using Enterprise Architecture Models and Bayesian Belief Networks for Failure Impact Analysis". In: *Int. Conference on Service-Oriented Computing*. Ed. by George Feuerlicht and Winfried Lamersdorf. Springer, 2009, pp. 339–350.

[Hol+12] Hannes Holm et al. "Automatic Data Collection for Enterprise Architecture Models". In: *Software & Systems Modeling* 13 (May 2012). DOI: 10.1007/s10270-012-0252-1.

[HSB22] Simon Hacks, Muhamed Smajevic, and Dominik Bork. "Using Knowledge Graphs to Detect Enterprise Architecture Smells". In: *EMISA 2022*. Ed. by Henrik Leopold and Henderik A. Proper. Bonn: Gesellschaft für Informatik e.V., 2022.

[IJ06] Maria-Eugenia Iacob and Henk Jonkers. "Quantitative analysis of enterprise architectures". In: *Interoperability of Enterprise Software and Applications*. Springer, 2006, pp. 239–252.

[Int] MEGA International. *MEGA International Enterprise Architecture*. [Online; accessed 28.11.2021] https://www.mega.com/en/product-enterprise-architecture. URL: https://www.mega.com/en/product-enterprise-architecture. (accessed: 28.11.2021).

[Jou+21] Hervé Jouin et al. *Archimate Tool Database-Plugin*. [Online;] https://github.com/archi-contribs/database-plugin. 2021. URL: https://github.com/archi-contribs/database-plugin. (accessed: 16.01.2022).

[JPT09] Henk Jonkers, Henderik Proper, and M Turner. "TOGAF 9 and ArchiMate 1.0". In: (Nov. 2009).

[Lag+13] Robert Lagerström et al. "Visualizing and measuring enterprise architecture: an exploratory biopharma case". In: *IFIP Working Conference on The Practice of Enterprise Modeling*. Springer. 2013, pp. 9–23.

[Lan+09] Marc Lankhorst et al. *Enterprise architecture at work*. Vol. 352. Springer, 2009.

[Lan+12] Philip Langer et al. "Model Transformation By-Example: A Survey of the First Wave". In: vol. 7260. Jan. 2012, pp. 197–215. ISBN: 978-3-642-28278-2. DOI: 10.1007/978-3-642-28279-9_15.

[Lan+16] Birger Lantow et al. "Towards a Classification Framework for Approaches to Enterprise Architecture Analysis". In: *Proceedings PoEM 2016*. Nov. 2016, pp. 335–343.

[Laz19] Ljubica Lazarevic. *Using Neo4j to explore your ArchiMate model*. [Online; accessed 16.01.2022] https://lju-lazarevic.github.io/ArchiMateNeo4j1.html. 2019. URL: https://lju-lazarevic.github.io/ArchiMateNeo4j1.html. (accessed: 16.01.2022).

[Laz21] Ljubica Lazarevic. *Using a Graph Database to Explore Your ArchiMate Model*. [Online; accessed 16.01.2022] https://medium.com/geekculture/using-a-graph-database-to-explore-your-archimate-model-df7bd63f65dd. 2021. URL: https://medium.com/geekculture/using-a-graph-database-to-explore-your-archimate-model-df7bd63f65dd. (accessed: 16.01.2022).

[LC20] José Antonio Hernández López and Jesús Sánchez Cuadrado. "MAR: a structure-based search engine for models". In: *MoDELS '20: ACM/IEEE 23rd International Conference on Model Driven Engineering Languages and Systems, Virtual Event, Canada, 2020*. Ed. by Eugene Syriani et al. ACM, 2020, pp. 57–67.

[Lea] LeanIX. *Enterprise Architecture Management (EAM)*. [Online; accessed 28.11.2021] https://www.leanix.net/en/products/enterprise-architecture-management. URL: https://www.leanix.net/en/products/enterprise-architecture-management. (accessed: 28.11.2021).

[Leh+20] Barry-Detlef Lehmann et al. "Towards the Identification of Process Anti-Patterns in Enterprise Architecture Models". In: *8th International Workshop on Quantitative Approaches to Software Quality in conjunction with the 27th Asia-Pacific Software Engineering Conference (APSEC 2020)*. Vol. 2767. CEUR-WS, Dec. 11, 2020, pp. 47–54.

[LH12] Olga Levina and Robert Hillmann. "Network-Based Business Process Analysis". In: *Proceedings of the Annual Hawaii International Conference on System Sciences* (Jan. 2012), pp. 4356–4365. DOI: 10.1109/HICSS.2012.447.

[Lib19] The Open Group Library. "ArchiMate® Model Exchange File Format for the ArchiMate Modeling Language, Version 3.1". In: Nov. 2019, p. 39. ISBN: 1-947754-39-3.

[Mum+19] Haris Mumtaz et al. "A survey on UML model smells detection techniques for software refactoring". In: *Journal of Software: Evolution and Process* 31.3 (2019), e2154.

[neo21a] neo4j. *Eigenvector Centrality*. [Online; accessed 23.12.2021] https://neo4j.com/docs/graph-data-science/current/algorithms/eigenvector-centrality/. 2021. URL: https://neo4j.com/docs/graph-data-science/current/algorithms/eigenvector-centrality/. (accessed: 23.12.2021).

[neo21b] neo4j. *PageRank Centrality*. [Online; accessed 23.12.2021] https://neo4j.com/docs/graph-data-science/current/algorithms/page-rank/. 2021. URL: https://neo4j.com/docs/graph-data-science/current/algorithms/page-rank/. (accessed: 23.12.2021).

[NSV15] David Naranjo, Mario Sánchez, and Jorge Villalobos. "PRIMROSe: A Graph-Based Approach for Enterprise Architecture Analysis". In: July 2015, pp. 434–452. ISBN: 978-3-319-22347-6. DOI: 10.1007/978-3-319-22348-3_24.

[ÖLR12] Magnus Österlind, Robert Lagerström, and Peter Rosell. "Assessing Modifiability in Application Services Using Enterprise Architecture Models – A Case Study". In: *Proceedings TEAR 2012*. Springer, 2012, pp. 162–181.

[OMG19] OMG. *ArchiMate® 3.1 Specification*. http://pubs.opengroup.org/architecture/archimate3-doc/. The Open Group, 2019. URL: http://pubs.opengroup.org/architecture/archimate3-doc/ (visited on 03/29/2021).

[PB17] Benedikt Pittl and Dominik Bork. "Modeling digital Enterprise ecosystems with ArchiMate: a mobility provision case study". In: *International Conference on Serviceology*. Springer. 2017, pp. 178–189.

[PV18] Murugaiyan Pachayappan and Ramakrishnan Venkatesakumar. "A graph theory based systematic literature network analysis". In: *Theoretical Economics Letters* 8.05 (2018), pp. 960–980.

[Ram+14] Andres Ramos et al. "Automated Enterprise-Level Analysis of ArchiMate Models". In: *Enterprise, Business-Process and Information Systems Modeling*. Ed. by Ilia Bider et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 439–453. ISBN: 978-3-662-43745-2.

[Rob+15] Stewart Robinson et al. "Conceptual Modeling: Definition, Purpose, and Benefits". In: Dec. 2015. DOI: 10.1109/WSC.2015.7408386.

[Sal+21] Johannes Salentin et al. *Enterprise Architecture Smells Catalog*. https://swc-public.pages.rwth-aachen.de/smells/ea-smells/. 2021.

[San+16] Alixandre Santana et al. "Combining network measures and expert knowledge to analyze enterprise architecture at the component level". In: *2016 IEEE EDOC Conference*. IEEE. 2016, pp. 1–10.

[SB21a] Muhamed Smajevic and Dominik Bork. "From Conceptual Models to Knowledge Graphs: A Generic Model Transformation Platform". In: *2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C) - Tools & Demonstrations Track*. ACM/IEEE. USA: IEEE Xplore Digital Library, 2021, pp. 610–614. ISBN: 978-1-6654-2484-4. DOI: 10.1109/MODELS-C53483.2021.00093. URL: https://publik.tuwien.ac.at/files/publik_297025.pdf.

[SB21b] Muhamed Smajevic and Dominik Bork. “Towards Graph-based Analysis of Enterprise Architecture Models”. In: *40th International Conference on Conceptual Modeling*. Ed. by Aditya Ghose et al. Springer. Springer, LNCS, 2021, pp. 199–209. ISBN: 978-3-030-89021-6. DOI: 10.1007/978-3-030-89022-3_17. URL: https://publik.tuwien.ac.at/files/publik_297029.pdf.

[Sch16] Anthony Schoonjans. “Social network analysis techniques in enterprise architecture management”. PhD thesis. PhD thesis, Ghent University, Ghent, 2016.

[Ser] ServiceNow. *ServiceNow named a Leader for 8th year in Gartner 2021 ITSM Magic Quadrant*. [Online; accessed 28.11.2021] https://www.servicenow.com/lpayr/gartner-mq-itsm.html. URL: https://www.servicenow.com/lpayr/gartner-mq-itsm.html. (accessed: 28.11.2021).

[SFM16] Alixandre Santana, Kai Fischbach, and Hermano Moura. “Enterprise architecture analysis and network thinking: A literature review”. In: *2016 49th Hawaii International Conference on System Sciences (HICSS)*. IEEE. 2016, pp. 4566–4575.

[SGR20] Shreya Srinivas, Asif Gill, and Terry Roach. “Analytics-Enabled Adaptive Business Architecture Modeling”. In: *Complex Systems Informatics and Modeling Quarterly* (July 2020), pp. 23–43. DOI: 10.7250/csimq.2020-23.03.

[SH20a] Johannes Salentin and Simon Hacks. *Enterprise Architecture Smells Prototype*. https://git.rwth-aachen.de/ba-ea-smells/program. 2020.

[SH20b] Johannes Salentin and Simon Hacks. “Towards a Catalog of Enterprise Architecture Smells”. In: *Entwicklungen, Chancen und Herausforderungen der Digitalisierung: Proceedings der 15. Internationalen Tagung Wirtschaftsinformatik, WI 2020, Potsdam, Germany, March 9-11, 2020. Community Tracks*. Ed. by Norbert Gronau et al. GITO Verlag, 2020, pp. 276–290.

[SHB21] Muhamed Smajevic, Simon Hacks, and Dominik Bork. “Using Knowledge Graphs to Detect Enterprise Architecture Smells”. In: *Proceedings of the 14th IFIP Working Conference, PoEM 2021, Riga, Latvia, November 24-26, 2021*. Springer International Publishing, 2021, pp. 48–63. ISBN: 978-3-030-91278-9. URL: https://publik.tuwien.ac.at/files/publik_297954.pdf.

[Sol+02] Rini Solingen et al. “Goal Question Metric (GQM) Approach”. In: Jan. 2002. ISBN: 9780471028956. DOI: 10.1002/0471028959.sof142.

[SS06] Hans-Jörg Schulz and H. Schumann. “Visualizing Graphs - A Generalized View”. In: vol. 9. Aug. 2006, pp. 166–173. ISBN: 0-7695-2602-0. DOI: 10.1109/IV.2006.130.

[SS15] Prince M Singh and Marten J van Sinderen. “Lightweight metrics for enterprise architecture analysis”. In: *International Conference on Business Information Systems*. Springer. 2015, pp. 113–125.

[SSS14] Girish Suryanarayana, Ganesh Samarthyam, and Tushar Sharma. *Refactoring for software design smells: managing technical debt*. Morgan Kaufmann, 2014.

[TH21] Benny Tieu and Simon Hacks. “Determining Enterprise Architecture Smells from Software Architecture Smells”. In: *23rd IEEE International Conference on Business Informatics Workshops (to be published)*. IEEE, 2021.

[Van09] Ermersj J. Van Sante T. *TOGAF 9 and ITIL V3. White Paper*. http://www.best-management-practice.com/. 2009.

[Ven+14] R. K. M. Veneberg et al. “Enterprise Architecture Intelligence: Combining Enterprise Architecture and Operational Data”. In: *2014 IEEE 18th International Enterprise Distributed Object Computing Conference*. 2014, pp. 22–31. DOI: 10.1109/EDOC.2014.14.

[VGM13] Marco Vicente, Nelson Gama, and Miguel Mira da Silva. “Using ArchiMate and TOGAF to Understand the Enterprise Architecture and ITIL Relationship”. In: vol. 148. June 2013, pp. 134–145. ISBN: 978-3-642-38489-9. DOI: 10.1007/978-3-642-38490-5_11.

[Woo+13] John Wood et al. “A framework for capturing the hidden stakeholder system”. In: *Systems Engineering* 16 (Sept. 2013). DOI: 10.1002/sys.21224.

[Zac87] John Zachman. “A framework for information systems architecture”. In: *IBM Systems Journal* 26 (1987), pp. 276–292.