The present work was submitted to
the Research Group
Software Construction

of the Faculty of Mathematics,
Computer Science, and
Natural Sciences

Master Thesis

Enterprise Architecture Debt Modelling

presented by

Yevheniia Kashperuk

Aachen, March 24, 2022

Examiner

Prof. Dr. rer. nat. Horst Lichter

Prof. Dr. rer. nat. Bernhard Rumpe

Supervisor

Peter Alexander, M.Eng.

Statutory Declaration in Lieu of an Oath

The present translation is for your convenience only.
Only the German version is legally binding.

I hereby declare in lieu of an oath that I have completed the present Master's thesis entitled

Enterprise Architecture Debt Modelling

independently and without illegitimate assistance from third parties. I have used no other than the specified sources and aids. In case that the thesis is additionally submitted in an electronic format, I declare that the written and electronic versions are fully identical. The thesis has not been submitted to any examination body in this, or similar, form.

Official Notification

Para. 156 StGB (German Criminal Code): False Statutory Declarations
Whosoever before a public authority competent to administer statutory declarations falsely makes such a declaration or falsely testifies while referring to such a declaration shall be liable to imprisonment not exceeding three years or a fine.

Para. 161 StGB (German Criminal Code): False Statutory Declarations Due to Negligence
(1) If a person commits one of the offences listed in sections 154 to 156 negligently the penalty shall be imprisonment not exceeding one year or a fine.
(2) The offender shall be exempt from liability if he or she corrects their false testimony in time. The provisions of section 158 (2) and (3) shall apply accordingly.

I have read and understood the above official notification.

Eidesstattliche Versicherung

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Masterarbeit mit dem Titel

Enterprise Architecture Debt Modelling

selbständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Aachen, March 24, 2022 (Yevheniia Kashperuk)

Belehrung

§ 156 StGB: Falsche Versicherung an Eides Statt
Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicher ung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt
(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.
(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.

Die vorstehende Belehrung habe ich zur Kenntnis genommen.

Aachen, March 24, 2022 (Yevheniia Kashperuk)

Acknowledgment

I want to thank all the people who have contributed to this thesis. With your help and support, I have reached this important milestone in my life.

Special thanks to my therapist who pulled me out of a time when I was too close to giving up.

Yevheniia Kashperuk

Abstract

The idea that different languages are spoken among IT and other departments is a typical challenge in organisations. Fellow employees come from a variety of backgrounds, have different levels of understanding, and occasionally even have conflicting goals, making alignment more difficult. Enterprise Architecture (EA) plays an important role in connecting IT objectives with business goals, potentially resolving business and IT misalignments issues by providing a common language for them and adding more value to companies. There are a lot of approaches to resolve the communication issue and make EA management more straightforward. But there is one concept in this domain that lacks attention.

Originally, the term "debt" was borrowed to IT from the finance domain. The concept of EA debt extends its focus to include business aspects. Enterprise Architecture debt is a metric that depicts the deviation of the currently present state of an enterprise from a hypothetical ideal state. To be able to communicate the severity of an EA debt to stakeholders, a management framework was designed. The main challenge in realising management activities is to integrate the employed documentation and communication approaches with the viewpoints and information interests of different stakeholders. And in this work, we try to resolve this issue and provide a tool that will align people in an enterprise providing a ground truth information point.

We argue that a possible solution is to communicate EA debt using a modelling approach. As a visualisation tool, modelling adds structure to the management workflow, provides a bigger picture as well as different levels of abstraction for identifying the elements of EA debt. Communicating debt through the use of models will reduce the gap between a problem and an architectural state as they describe complex systems at multiple levels of abstraction and from a variety of perspectives. Therefore, modelling support is needed to specify, document, communicate and reason about EA debt.

In this thesis, we introduce the extension for modelling notations which includes the EA debt specific elements to capture EA debt and make it easier to document and communicate it between stakeholders. The study begins with a literature review to collect domain insights and knowledge needed to define use cases of EA debt modelling. These use cases serve as proof of why the EA field will benefit from the modelling tool. Based on this information, we identified six core concepts of EA debt and built a meta-model that represents their relations with each other. Through interviews with experts and practitioners in the field, the developed meta-model was evaluated to ensure that all aspects of EA debt are covered. The expert comments gathered during the evaluation proved that, together with the proposed solution, the EA debt management process would provide more value to stakeholders allowing them to be more confident with their decisions.

Contents

List of Tables

List of Figures

1. Introduction

Contents

The following chapter introduces a motivation of the study followed by purpose, research questions and delimitations.

1.1. Motivation

Enterprise Architecture (EA) has shown a rise in terms of interest in recent years [Lan+10]. Many organisations now acknowledge EA as a critical discipline and practice because it helps them face ongoing and disruptive change. There are several enterprise architecture initiatives, such as The Open Group Architecture Framework (TOGAF) [Tog], the Zachman Framework [SZ92], Integrated Architecture Framework (IAF) [Iaf], and Design & Engineering Methodology for Organisations (DEMO) [Die01]. The use of models in the form of diagrammatic descriptions of information systems and their environments is widely known. EA models, on the other hand, are not just for descriptive purposes. They may also be used to forecast decision behaviour and consequences. However, existing techniques do not take into account a wide range of situations, i.e., they are not flexible to varied contexts and goals. Essentially, EA models aid various stakeholders in the organisation in documenting and thus understanding the complex enterprise, analysing the properties of current and potential future scenarios, planning and designing future scenarios and the path to get there, and communicating the current and future state of affairs to other stakeholders. Furthermore, by focusing on the needs of decision-makers – modelling for specific goals rather than modelling for the sake of modelling – EA model can be used as a strong decision-support tool [JE07]. Moreover, modelling might help to overcome a serious obstacle to rational decision-making, which is uncertainty. An effective EA model should be able to capture ambiguities in assessment theory, system design, or data quality, allowing for better decision-making and risk management. While much of today available literature and tools address EA modelling quite well, our focus is on EA debt.

The EA debt was first defined by Hacks et al. [Hac+19] as “a metric that depicts the deviation of the currently present state of an enterprise from a hypothetical ideal state”.

Since business and IT representatives potentially have different mindsets and different goals, it is believed that EA debt plays an important role to provide a common language for them. Here, models can help to address the need of covering the information interests of different stakeholders. Including them as a part of the documentation will provide an easy-to-read summary of the situation, which might be useful for parties who are not deeply involved in the workflow, but still need to be informed or regularly updated. Debt topic is always a part of the management process, even if some organisations address it as "temporarily drawbacks" or "shortcuts". So it is important to properly specify, document, and communicate it. Of course, there are existing modelling languages that support enterprise architecture modelling. But the problem here is that none of them fully support debt modelling. This is why we see a possibility to close this gap and improve EA management by presenting an EA debt modelling approach.

1.2. Goal

The aim of this study is to explore possible situations in which having an EA debt modelling toolbox would be useful and to prove why this is so. A key contribution will be the development of a modelling approach to capture and communicate various aspects of EA debt.

The outcome of this work will help future researchers and practitioners to get an implication on how to conduct research in the EA debt domain and come up with ways of further developing ways of adopting EA and EA debt modelling in real-world scenarios.

1.3. Research questions

For the purpose of designing the highly functional modelling notation, we formulate following our research questions.

1. What are the use cases of EA debt modelling?
   a) What are the current approaches to debt modelling?
   b) What knowledge can be adapted to current work?
2. What are the core elements of EA debt that are needed to support the use cases?
3. How to formalise the core elements into the modelling notation?

The first sub-question RQ1.a focuses on different approaches to debt modelling described in the publications. Analysing the available work in the domain in RQ1.b (focused on, but not limited to, enterprise architecture) will provide an understanding of what knowledge and how it can be transformed into the EA debt topic. The results of these two questions will serve as an input to RQ1 to define use cases of EA debt modelling. Question R2 concentrates on identifying key concepts of debt and information needed to have a complete idea of the debt-related situation. An answer to RQ3 will be an investigation of how previously defined notions can be visualised to provide a meaningful image of the EA debt situation.

1.4. Delimitations

Due to the fact that EA debt is a relatively new field, only a few people can be considered domain experts. But even with resources available, it might be complicated to maintain a close and frequent collaboration because of practitioners' workload in terms of getting feedback and availability to participate in evaluation sessions.

Another limitation connected to the newness of the EA debt is that there is not much specific research available. Which is a crucial point in analysing state-of-the-art in the domain or looking for existing solution alternatives. So, various related topics have to be investigated and analysed on how this study can extend prior work. It can introduce bias due to the slightly different focus of these studies.

2. Theoretical Background

Contents

This chapter presents a theoretical foundation of the study. After a brief introduction to Enterprise Architecture, the concept of Technical Debt is provided, followed by the explanation of the Enterprise Architecture Debt, which is a concept where Technical Debt is extended to the Enterprise Architecture domain.

2.1. Enterprise Architecture

As a motivation for his SLR, Saint-Louis stated that despite the increased interest among researchers and practitioners, several studies reported a lack of common understanding concerning EA [SLML17]. The absence of a common understanding may foster confusion and conflicts concerning the meaning of "enterprise architecture" as well as the role of professionals practising enterprise architecture. As a result of his study, one hundred forty-five EA definitions found in the literature were decomposed into several parts based on concepts from the field of terminology. The findings show there are many divergences between EA definitions and the nature of some of them are significant. Therefore, it is reasonable to regard EA as a set of artefacts, which are aggregated. Kappelman pointed out that "The 'enterprise' portion of EA is understood by some as a synonym to 'enterprise systems', yet by others as equivalent to 'business' or 'organization'" [Kap+08]. Even less uniform is the understanding of the meaning of 'architecture'. The most common understanding of the term is a collection of artefacts (models, descriptions, etc.) that define the standards of how the enterprise should function or provide an as-is model of the enterprise".

Despite all possible definitions, EA helps to face "ongoing and disruptive change" by attempting to align IT and business strategy [Hac+19]. Its main goal can be defined as to lay out how information, business, and technology interact. As EA is driven by

the organisation's business requirements, it helps to align teams and provide a unified vision inside the organisation. It has become a top priority for businesses striving to keep up with new technologies like the cloud, internet of things, machine learning, and other upcoming trends that will inspire digital transformation.

Enterprise architecture is unique to every organization, however, there are some common elements. Since Stephen Spewak's Enterprise Architecture Planning in 1993 [SH93] it has been normal to divide enterprises architecture into four architecture domains (Figure 2.1).

The four commonly accepted domains of enterprise architecture are:

- Business architecture domain – describes how the enterprise is organizationally structured and what functional capabilities are necessary to deliver the business vision.
- Application architecture domain – describes the individual applications, their interactions, and their relationships to the core business processes of the organization.
- Data architecture domain – describes the structure of an organization's logical and physical data assets and data management resources. Knowledge about your customers from data analytics lets you improve and continuously evolve business processes.
- Technology architecture domain – describes the software and hardware needed to implement the business, data, and application services. Each of these domains have well-known artifacts, diagrams, and practices.

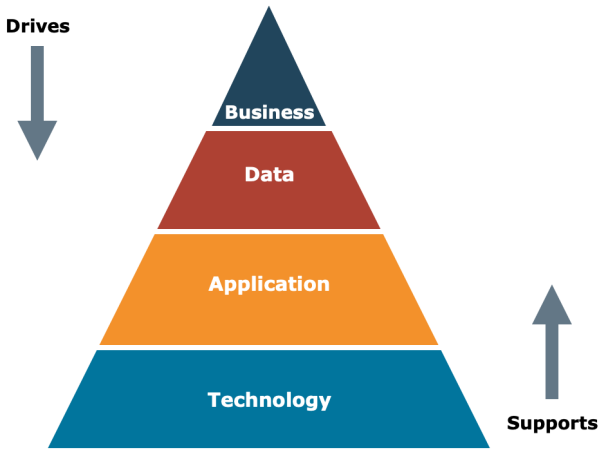

Figure 2.1.: Layers of Enterprise Architecture

For many years, it has been common to regard the architecture domains as layers, with the idea that each layer contains components that execute processes and offer services

to the layer above. Many EA frameworks combine data and application domains into a single information system layer, sitting below the business and above the technology (the platform IT infrastructure).

2.2. Enterprise Architecture Modelling

According to [Pro+18], there have been several language engineering efforts in the context of EA modelling. In the SLR done by Zhou et al. [Zho+20] they extracted EA visualisation methods used or recommended by the researchers 2.1. For our work, we will focus on the first four of them, namely, ArchiMate, UML, BMM, and BPMN, as being most common and widely mentioned in the literature. Each framework possesses different strengths and weaknesses which are discussed further.

EA model notations	Number of Papers
ArchiMate	41
UML	13
BMM (Business Motivation Model)	11
BPMN	5
CySeMoL	3
SoaML	3
SysML	2
GSN	2
CMMN	2
URN (User Requirement Notation)	2

Table 2.1.: Mainly used modelling languages in the selected studies

2.2.1. ArchiMate

ArchiMate [LPJ10], an open standard of The Open Group, defines concepts for describing architectures at the business, application, and technology layers, as well as the relationships between these layers. Thus, it addresses the ubiquitous problem of business–IT alignment. Additionally, being an industry standard ArchiMate provides such advantages as the availability of know-how and resources, trainings availability on the market, the availability of best practices that have been developed by others, and support by a variety of different tools. ArchiMate originally results from a public/private research project, a cooperation of companies, universities and research institutes.

ArchiMate focuses on the modelling of extensional and intentional properties of an enterprise, in terms of informational, behavioural and structural architecture elements. Extensional properties model a system from an external perspective, e.g. the products and services that are offered. Intentional properties model the system from an internal perspective, e.g. how the products and services are supported by processes and applications.

The modelling framework that underlies the ArchiMate language decomposes an enterprise along two dimensions: layers, which represent successive abstraction levels at which an enterprise is modelled, and aspects, which represent different concerns of the enterprise that need to be modelled. The layer dimension distinguishes three main layers:

- business layer, which offers products and services to external customers that are realised in the organisation by business processes;
- application layer, which supports the business layer with application services that are realised by (software) application components;
- technology layer, which offers infrastructural services (e.g. processing, storage and communication services) that are needed to run applications, and are realised by computer and communication devices and system software.

The aspect dimension distinguishes the following modelling aspects:

- structure aspect, which represents the actors (systems, components, people, departments, etc.) involved and how they are related;
- behaviour aspect, which represents the behaviour (e.g. processes and services) that is performed by the actors, and the way the actors interact;
- information aspect, which represents the problem domain knowledge that is used by and communicated between the actors through their behaviours.

The structuring into dimensions allows one to model an enterprise from different viewpoints, where a viewpoint is characterised by one's position along each dimension. A viewpoint represents a certain perspective on the enterprise that is of interest to one or more stakeholders. A stakeholder typically focuses on a (small) range along each of the dimensions. The intersection of these ranges spans a viewpoint. A viewpoint may span multiple or only part of a layer or aspect. Furthermore, depending on the choice of viewpoints, they may (and often will) overlap. Each viewpoint comprises a number of concepts that are used to model an EA covering the levels of abstraction and aspects represented by that viewpoint. Accordingly, overlapping viewpoints may comprise overlapping concepts. In order to define, maintain and apply concepts for EA modelling in a structured and consistent way, these concepts are organised in orthogonal, i.e. non-overlapping 'viewpoints', called domains. Each domain represents a set of concepts that is used to model systems from a particular viewpoint.

On the disadvantages side, people state that ArchiMate is quite complicated, not clear, not easy to read and not easy to understand. This opinion seems to be supported by every new release of ArchiMate, as every release adds new complexity to the framework and new elements to the notation. The latest version has about 60 different elements that can be used for modelling. While some say that those are too many elements, others say that they are not sufficient to describe everything they would like to describe. Another aspect that is often considered in today´s IT world is whether the tool, approach or framework supports an agile methodology. For ArchiMate, whose main advantage is to provide structure and illustrate situations clearly, this is not the case.

2.2.2. BPMN

The Business Process Modeling Notation (BPMN) [Gro11] is a standardised, graphical notation used to model business processes and workflows. The main goal of BPMN is to provide notation that is truly understandable by all enterprise users, from business analysts who create initial sketches or processes, through developers in charge or setting up the technology that will run these processes, right up to enterprise users who will manage and supervise the processes.

BPMN specifies a single Business Process Diagram (BPD). This diagram has two main advantages. First, it is easy to use and understand. You can use it to quickly and easily model business processes, and it is easily understandable by non-technical users (usually management). Second, it offers the expressiveness to model very complex business processes, and can be naturally mapped to business execution languages.

To model a business process flow, you simply model the events that occur to start a process, the processes that get performed, and the end results of the process flow. Business decisions and branching of flows is modeled using gateways. A gateway is similar to a decision symbol in a flowchart.

The BPMN standard covers a lot of ground. It spans 500 pages and includes more than 100 graphical process elements. This makes learning and adapting to BPMN rather challenging. Furthermore, if BPMN elements or the semantics of BPMN diagrams are not learned comprehensively, users may interpret them in different ways, leading to inaccurate conclusions.

Different BPMN vendors implement the execution of BPMN diagrams in different (i.e. non-standardized) methods, despite its established execution semantics. This makes it difficult to share BPMN diagrams between tools and limits modelers to particular products or providers. Furthermore, the majority of BPMN tools only provide only a subset of BPMN elements, thus limiting support for BPMN diagram execution.

2.2.3. BMM

The Business Motivation Model (BMM) [Gro08] provides a structure of concepts for developing, communicating and managing business plans. The concepts can be used to model (i) the factors that motivate a business plan, (ii) the elements that constitute the business plan and (iii) the relationships between these factors and elements. The BMM has been developed by the Business Rules Group (Business Rules Group 2008) and has been adopted as an OMG standard in 2005.

The central notion of the BMM is motivation. An enterprise should not only define in its business plan what approach it follows for its business activities, but also why it follows this approach and what results it wants to achieve. Figure 5 depicts an overview of the BMM. The following three major parts are distinguished:

- Ends, which describe the aspirations of the enterprise, i.e. what the enterprise wants to accomplish;

- Means, which describe the action plans of the enterprise to achieve the ends, and the capabilities that can be exploited for this purpose;

- Influencers, which describe the assessment of the elements that may influence the operation of the enterprise, and thus influence its ends and means.

2.2.4. UML

UML [Gro09] was standardised in 1997, and a major new version was published in 2005. UML groups together a large number of modelling techniques that were previously scattered among different domains (entity relationship, object model, state diagram, sequence diagram. process modelling, etc.). It is widely accepted and used in the modelling of software systems. UML enables data to be modelled through class diagrams. Behaviour is modelled through object modelling (object behaviors, operations, etc.) and the support of sequence diagrams, state diagrams, and activity diagrams. Systems and architectures are also modelled using the concept of components and component assembly techniques. UML is highly flexible, allowing customisation of the modelling elements and interactions in a diagram specifically to suit the domain or technologies of the organisation.

A term coined by George Fairbanks, 'architecture-indifferent design' [FG10] is a situation where UML is considered unnecessary. At its core, an architecture-indifferent design refers to a software architecture that is simple and basic, and does not need any complex diagrams to represent or explain the design. If the firms lay more emphasis on formal coding, and there is a prevalent culture of minimal design documentation, UML is regarded unnecessary.

2.3. Technical Debt

Technical debt (TD) is a metaphor reflecting technical compromises that can yield short-term benefits but may hurt the long-term health of a software system. Introduced by Cunningham [Cun92], this metaphor was initially concerned with software implementation (i.e., at code level), but it has been gradually extended to software architecture, detailed design, and even documentation, requirements, and testing [LAL15]. Although the context of TD is still limited to the technological aspects.

A software project can both benefit from and be harmed by TD. Intentionally incurred TD can also be used strategically. In some cases, by delaying certain maintenance tasks or doing them quickly and less carefully, software managers can trade-off software quality with productivity. For instance, incurring TD can speed up the development of new features, thus helping the company move ahead of competition [GS11]. On the other hand, TD can also be incurred unintentionally, meaning that the project manager and development team are not aware of the existence, location, and consequences of the TD. If left invisible and unresolved, TD can be accumulated incrementally, which in turn results in challenges for maintenance and evolution tasks [LAL15].

In 2009 Martin Fowler introduced the Technical Debt Quadrant (Figure 2.2) and defined four types of debt: Reckless-Deliberate, Prudent-Deliberate, Reckless-Inadvertent, and Prudent-Inadvertent. He argues that "useful distinction is not between debt or non-debt, but between prudent and reckless debt" [Fow09]. Further differentiation of TD to deliberate and inadvertent provides an additional level of insights. TD will exist even in best-planned software projects, and therefore it is important not to be more reckless than necessary and not write crummy code on purpose.

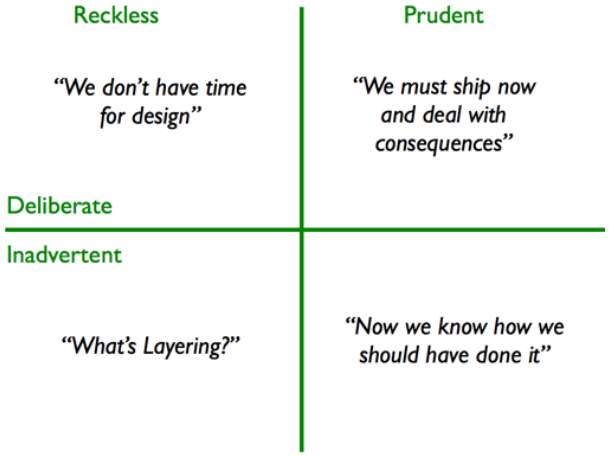

Figure 2.2.: Technical Debt Quadrant [Fow09]

2.4. Enterprise Architecture Debt

However, the concept of technical debt that particularly focuses on technical aspects demonstrates a lack of attention to attaining a holistic perspective to address the alignment between business and IT aspects. While enterprise architecture management (EAM) is gaining significant attention as a management instrument in business and IT [Hac+19]. By adapting the TD concept in the EA domain, a new metaphor, providing a holistic perspective, has been proposed.

Hacks et al. [Hac+19] define EA Debt as "a metric that depicts the deviation of the currently present state of an enterprise from a hypothetical ideal state". EA Debt arises when debt is taken in an artefact, which an EA consists of. It means that an element is not implemented or executed optimally in relation to the supposed ideal situation. Taking debt in a low hierarchy can be helpful and pay off, but it has to be "repaid" in accordance with business-related goals. Otherwise, the whole EA would rely on that debt and use faulty or considered bad artefacts. It entails a high risk of additional debt and hinders development. EA Debt is further increased by bad interfaces or bad interoperability and different priorities of stakeholders, not conforming with an EA that is considered good by evaluation approaches.

However, despite its importance and widespread presence, as of today, our knowledge

of EA debt is still incomplete. It is still an open question how to accurately identify, monitor, and manage EA debt.

There is a very limited number of studies that present possible frameworks for strategically managing EA-debt-related issues. One example is work by Alexander et al. [Ale+20] which presents a framework for EAD Management (EADM) that is built by adopting TDM concepts into the EA domain. The framework defines the following nine key activities (Figure 2.3):

- The identification activity focuses on recognising signs of possible EA debts.
- The next step is the collection of evidence, based on which suspicions of EA debts can be raised, confirmed/dispelled, and organised for further management activities.
- The assessment activity quantifies the business consequences of the identified EA debts.
- The prioritisation activity uses assessment results as a basis for reasoning. Concerning business goals and circumstances, EA debts are ranked based on a set of criteria such as the overall impact and mitigation intricacy, thereby helping to come up with a feasible and effective mitigation strategy.
- The monitoring activity addresses the continuous changes in EA due to changing business requirements, technical innovation, or reorganisation.
- Next, either the repayment of existing EA debts or the prevention of a worsening EA debts situation can be used as mitigating strategies. Through these activities, alternative scenarios in repaying/preventing high-priority EA debts are devised, and the best activity is selected based on a specific principle.
- Finally, the documentation and communication activities sustain the flow of EA debt knowledge among the involved stakeholders.

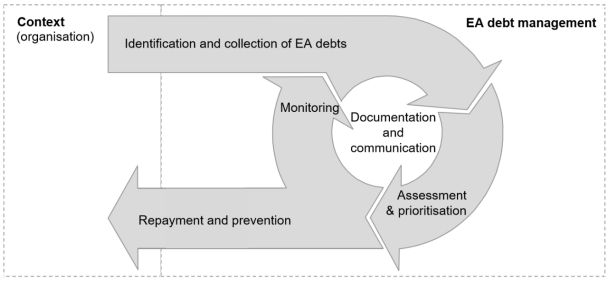

Figure 2.3.: EADM framework overview [Ale+20]

3. Literature Review

Contents

This chapter describes the methodology and results of the literature review conducted to explore the state of the research in the EA domain. The outcome of the data collection, with respect to the investigated research questions, is divided into three groups following the viewpoints defined by [Gro13].

3.1. Systematic literature review

To collect and analyse existing studies in related fields, a systematic literature review (SLR) was chosen as the research methodology based on guidelines described by Kitchenham [KC07]. The author defines SLR as "a means of identifying, evaluating and interpreting all available research relevant to a particular research question, or topic area, or phenomenon of interest". The SLR includes the suggested following steps:

1. Background: the rationale for the survey.
2. Research questions: to achieve the objectives.
3. Primary study search strategy.

   A searching strategy process should include:

   i. Defining a searching term (query).
   ii. Defining the target for the searching term.
   iii. Selecting different data sources to identify candidate publications.

4. Study selection criteria: to govern the selection of primary studies.

   There are two types of criteria used in the research selection process: inclusion and exclusion. The criterion should be based on the research questions and be applied after the full text has been retrieved.

5. Study selection procedures: once the potentially relevant primary studies have been obtained, they need to be assessed for their actual relevance.

   The study selection includes the following steps:

   i. Retrieving the initial list of publications by manual search in databases with the defined search string.

   ii. Filtering retrieved list using the same search query with title and abstract-based selection in Citavi Reference Manager [Cit].

   iii. Applying inclusion criteria with a full-text-based selection.

   iv. Final list of publications as a result.

6. Study quality assessment: to assess the quality of primary studies in addition to general inclusion/exclusion criteria.

7. Data extraction strategy: to design data extraction forms to accurately record the information researchers obtain from the primary studies.

3.2. Background

The goal of this review is to gain knowledge primarily on EA debt modelling and management (challenges and solutions) as well as on related topics (e.g., Architectural Technical Debt (ATD)) to analyse possible modelling use-cases, look for ideas, and gain insights from existing solutions.

3.3. Research questions

To reach the goal of this review, we need to answer the RQ 1.a and 1.b. The outcome of this chapter will be a base for defining debt modelling use cases, that is, answering the RQ1.

3.4. Primary study strategy

Since EA debt is a relatively novel concept, it is inevitably a field with limited access to current solutions literature. Although, there are many parallels and linkages between the EA and EA debt domains and software development. Therefore, we decided to search for related works in the field of technical debt and ATD as well. To increase the likelihood to find publications addressing modelling approaches of the debt, the target of the searching term is defined to search by title, abstract, and full text.

The search query contains the following keywords: ***architect* AND debt AND model****.

The searching terms are combined using a Boolean AND operator, which entails that publication needs to include all defined terms. To catch terms like "architecture" or "architectural" as well as "model" or "modelling", the asterisk character * is used, known as a wildcard, to match one or more inflected forms of the searching term.

Study searches were performed in five main electronic databases, as suggested in [LAL15]: IEEEXplore, ACM Digital Library, Science Direct, ISI Web of Science, Scopus. Additionally, dblp, IET, Wiley, and Springer were included as a part of the initial research. Table 3.1 contains the exact search string on each database together with the filters applied.

3.5. Study selection criteria

The inclusion criteria are the following:

1. A publication should define or discuss debt modelling issues in the context of technical or enterprise architecture.
2. Publications should be peer-reviewed, i.e., published in journals or conference proceedings.
3. Only publications written in English is included.
4. Time period: from 1992 (the year of the original definition of the TD metaphor [Cun92]) to 2021 (the year of the beginning of this work).
5. Document available for free online with university access in the selected digital libraries in a downloadable format (i.e.: .pdf, .doc).

Here, the exclusion criteria are the opposite of the admission criteria, and include, for example, publications written in any language other than English. Articles with a different publication date, document type, or topic than those indicated in the prior inclusion criteria, were also excluded. Content-wise, papers not describing debt modelling were excluded.

3.6. Selection procedure

After applying the search query to selected databases, the initial list of literature containing 3668 results was obtained. The result of performing the title and abstract-based filtering procedure using the same search query was the list of 24 publications. Finally, applying the inclusion criteria to the full text proved that each article is relevant to the analysis, the final list of 24 publications was confirmed. Additionally, selection results have been reviewed by another person to ensure that all rules have been followed and collected papers are relevant to this work. A complete list of selected studies can be found in the Appendix A.

Database	Search string	Filters
IEEEXplore	("All Metadata":architect*) AND ("All Metadata":debt) AND ("All Metadata":model*)	Year: 1992-2021; Journals; Conferences
ACM Digital Library	[All: architect*] AND [All: debt] AND [All: model*] AND [Publication Date: (01/01/1992 TO 31/05/2021)]	Year: 1992-2021
Science Direct	architect AND debt AND model	Year: 1992-2021; Article type: Review articles, Research articles; Subject areas: Business, Management and Accounting, Computer Science, Engineering
ISI Web of Science	architect* AND debt AND model*	Year: 1992-2021
Scopus	TITLE-ABS-KEY (architect* AND debt AND model*) AND PUBYEAR > 1991 AND PUBYEAR < 2022	Year: 1992-2021; Subject area: Computer Science, Engineering, Business, Management and Accounting, Decision Sciences; Document type: Conference Paper, Article; Source type: Conference Proceeding, Journal; Language: English
dblp	architect* AND debt AND model*	–
IET	architect* AND debt AND model*	Year: 1992-2021; Content type: Conference Paper
Wiley	"architect*" anywhere and "debt" anywhere and "model*" anywhere	Year: 1992-2021; Publication type: Journals; Subjects: Computer Science, Business and Management
Springer	architect* AND debt AND model*	Year: 1992-2021; Content type: Article; Discipline: Computer Science, Business and Management; Language: English

Table 3.1.: Search criteria applied to databases

3.7. Quality assessment

Since SLR is not the main goal of this study, the quality assessment step was excluded from the scope of work. As the number of selected studies is not high, we will assume that the quality of the data items to be extracted is relatively high and every publication provide a valuable contribution to the review.

3.8. Data extraction

Mendeley Reference Manager [Men] was used for further analysis of the collected articles. All the papers were read and analysed in terms of answering RQs. They were divided in three groups based on The Open Group framework for the definition and classification of viewpoints and views [Gro13]:

1. Designing – Design viewpoints support architects and designers in the design process from initial sketch to detailed design.

2. Deciding – Decision support views assist managers in the process of decision making by offering insight into cross-domain architecture relations, typically through projections and intersections of underlying models, but also by means of analytical techniques.

3. Informing – These viewpoints help to inform any stakeholder about the enterprise architecture, in order to achieve understanding, obtain commitment, and convince adversaries.

This classification was chosen to group models that serve the same purpose and display similar content in terms of presenting information to stakeholders. Furthermore, grouping models by viewpoints will ease the process of identifying debt modelling use cases (the next step in this work), as it will provide an understanding of the purpose and key functions of each model, as well as what stakeholders and how they will benefit from such a model.

Data was collected using the form shown in Table 3.2, including predefined Data Collection Variables. This enabled recording and tracking of full details of each surveyed publication. The first data collection variables [D1]–[D3] are primarily due to the demographic characterization of the study. The designed modelling approach in the research presented by variable [D4] provides information when processing RQ1.a, likewise [D5] reports the objectives addressed by the proposed method, and [D6] synthesizes the requirement to the model. To answer RQ1.b, variable [D7] investigates possible modelling restrictions, and [D8] focus on information used as a theoretical background for the selected study.

Variable	Data Item Name	Description	Relevant RQ
D1	Author	List of authors of the study	None
D2	Title	The title of the study	None
D3	Venue	The name of the publication venue of the study	None
D4	Debt modelling approach	The description of the modelling approach	RQ1.a
D5	Modelling objectives	Why the designed approach is needed	RQ1.a
D6	Modelling requirements	What requirements were defined for a modelling language	RQ1.a
D7	Limits	The limits on modelling that were identified in the study	RQ1.b
D8	Supporting knowledge	Information used as a background knowledge	RQ1.b

Table 3.2.: Data collection variables and their purpose

3.9. Results

The section is divided into three different categories accordingly to the previously mentioned classification framework [Gro13]. Thus, papers were analysed based on different viewpoints supported by each model. We start with models that support the designing viewpoint, followed by deciding and informing viewpoints. An additional remark is that even though not all the presented studies directly connect to the EA debt, their results can be adapted in the context of the EA debt domain, as we will see in the following chapter 4.

3.9.1. Designing

Metis [Met], System Architect [Sys], and Aris [Sch12] are just a few of the enterprise architectural software products on the market. These tools generally focus on the modelling of architecture whereas the analysis functionality is generally limited to performing an inventory or to sum costs over the modelled architecture. A number of studies extend the existing approaches to cover more areas of EA.

Buschle et al. [S01] present an enterprise architecture software tool, which not only provides the functionality to model enterprise architectures but also supports their analysis. The tool consists of two main components. The theory relevant to analysing a certain system quality, such as data quality or modifiability, is modelled in the first component. This can be considered as the definition of a language that is specifically designed to express a specific element, such as cyber security. The application of the theory to evaluate a specific enterprise architecture is supported by the second component. This is accomplished by modelling the "as-is" or "to-be" architecture of the enterprise. It is possible to determine how the architecture meets the requirements described by the theory using the models that have been built. Since the same language may be used to describe several architecture instances, the two-component architecture encourages the reuse of the created theory. The developed tool uses the Probabilistic Relation Models (PRM) formalism which enables uncertainty to be considered.

Engelsman et al. [S02] state that little or no attention is paid to represent explicitly the motivations or rationale, i.e. the why behind the architectures in terms of goals and requirements. They introduce the language called ARMOR which is based on the existing requirements modelling languages and aligned with the standard enterprise modelling

language ArchiMate. ARMOR extends the ArchiMate framework with the motivation aspect. They demonstrate how to realise traceability of stakeholder concerns to the architectural elements. This traceability is realised through goal refinement and providing means to integrate this into the architecture domain. With ARMOR it is possible to model and refine strategic goals and policies found in business plans. Goal refinement allows linking this business context to the new architectural elements, thus realising traceability. Through capturing these links, it becomes possible to reason about the effects of changing goals on the EA. Through following links in the requirements domain, the EA domain and their integration, we can derive which architectural elements are affected by a change in a high-level goal. Furthermore, ARMOR can be used to support stakeholders in reasoning about conflicting interests and solutions. Visualising the effects of conflicting goals helps to understand what are the effects of these conflicts on the EA and leads to dropping or changing certain goals by certain stakeholders.

According to Izurieta et al. [S03], if our goals are to increase the adoption of modelling among regular practitioners and to reduce overall amounts of TD in software, we must adopt a new taxonomy or complement an existing one with concepts that define TD at the modelling level during software specification. A taxonomy will help researchers to identify new measures that will allow them to estimate potential debt before moving forward with implementation. Authors believe that reducing Model-Driven Technical Debt (MDTD) will reduce TD in implemented systems, hence increasing overall system quality and decreasing maintenance costs.

3.9.2. Deciding

Guo Y. et al. [S04] adopts a portfolio-based approach from the financial domain to help software managers make informed decisions. The decision will be made, following the process that starts with the extraction of all technical debt items, followed by adjustment of the estimates for these items and the addition of a constraint to the portfolio approach to ensure no partial holding of any technical debt items. After a preferred risk level was set, the model generates the optimal portfolio of the technical debt items. Although, this method faces some limits: the model's assumptions do not match the real market situation; estimating the standard deviation of the return on technical debt still relies on human experience more than historical data.

Wagter R. et al., in their work [S05], present a framework that allows enterprises to make their coherence explicit, thus enabling them to govern their coherence. In general terms, the Enterprise Coherence Framework consists of a set of so-called cohesive elements and cohesive relationships between them. The overall level of cohesion within an actual enterprise is determined by the explicitness of the cohesive elements, and quality/consistency of the cohesive relationships, in this enterprise. This also allows enterprises to govern their cohesion, in particular by guarding the cohesive relationships.

One part of the research is focused on cloud-based platforms, providing models for prediction, estimation and quantification of the technical debt in various scenarios. One of the proposed frameworks comprises models, guidelines, tools and calculators to support the decision-making process, extending IBM Unified Method Framework and TOGAF

[S06]. Two other related works present a mathematical-based approach to provide the architect with the insights into managing the value of the structure and its utilities in supporting changes in Quality of Service (QoS) [S07] and to support the decision-making process from both a technical perspective and a financial perspective [S08].

A number of studies are using Design Rule Space based on design rule theory as a way to model and analyse debt claiming that the DRSpace approach provides one piece of this vision—a means by which a project manager or architect can continuously monitor product quality, comparing it to project, organisation, or industry norms [S09]. This technique can automatically identify debts, measure their maintenance consequences, model their growth [S10] [S11], and quantify them and the expected pay-back for refactoring these debts [S12]. Titan tool chain used in those studies helps to manage architecture debt: by tracking the architecture roots, by tracking architecture flaws and the project's decoupling level [S09].

In their work, Izurieta et al. [S13] investigate an approach that uses Common Weakness Scoring System scores relevant to architectural decisions to help rank TD issues associated with security weaknesses. They use the Quamoco quality model and static analysis tool to provide a relative ranking of weaknesses that help practitioners identify the highest risks in an organisation with the potential to impact TD.

Perez et al. [S14] propose REBEL, a semi-automated model-driven approach to manage ATD's lifecycle. REBEL is based on natural language processing, machine learning and model checking techniques on heterogeneous project artefacts to identify, measure and track the impact produced by the consciously injected ATD and its repayment strategy on the other architectural decisions. This proposal focuses on ATD at the architecture level only without considering source code.

3.9.3. Informing

An SLR conducted by Besker et al. [S15] provides a new and comprehensive understanding of ATD and raise awareness about what challenges ATD are surrounded by. The findings showed that there is wide agreement in the reviewed literature that ATD is of primary importance. The key contribution of this paper of interest for the current work is a novel descriptive model of ATD. This model summarises all the findings and allows improved identification of ATD and associated negative consequences and corresponding architectural technical debt management (ATDM) activities. The model illustrates ATD in a unified and comprehensive way by exploring different aspects and relationships, which are considered particularly valuable for managing and raising awareness about ATD. The model reveals that all categories of ATD (as debt) are related to the challenge of complexity and furthermore that all challenges are related to maintenance and evolvability. This model can help several different stakeholders within the software life-cycle process to be better and more informed, manage the software, with the goal of raising the success rate and lowering the rate of negative consequences.

Li et al. [S16] argue that ATD caused by decisions is still not effectively managed. In their work, they present an initial attempt to tackle this problem through the following:

- An ATD conceptual model. In this model, the core concept is the ATD item which acts as the basic unit to record ATD. Additionally, they present a template for documenting an ATD item, in which most elements are adopted from a defined conceptual model.

- An ATDM process applying the proposed conceptual model. They applied the proposed management process to managing ATD within the general architecting process in Hofmeister et al. [Hof+07].

Authors state on account of that a software architecture can be considered as a set of architecture decisions [JB05] and therefore, the architecting process can be regarded as a decision-making process. The objective of this approach is to make the architecture decision-making process easier by managing ATD, thereby assisting architects in making appropriate and well-founded judgments and ensuring that the ATD of a system remains controllable. Li and co-authors believe that their contribution to this end can facilitate optimal decision making in architecture design and achieve a controllable and predictable balance between the value and cost of architecture design in the long term.

The study by Saat J. et al. [S17] describes a series of possible situations and proposes a meta-model that serves as a kind of best practice template for the modelling efforts of enterprises finding themselves in each of the described situations. A core meta-model is described as a UML class diagram. It includes the entities and entity relations that are related to the defined attributes. The same approach is used by Martini and Bosch [S18]. The authors present a model which should be used as a guideline for practitioners to help recognise the presence of dangerous classes of ATD items and visualise the relationships between the certain key phenomena and their effects, so-called contagious debt and vicious circles.

A big part of the research focuses on debt identification. Several various methods were proposed to develop predictive models. For example:

- A framework proposed by Tommasel [S19] has social networks analysis at its core.

- Shahbazian et al. [S20] presented a method for automatically detecting architecturally significant issues and classifying them by applying the two architecture recovery techniques: Algorithm for Comprehension-Driven Clustering and Architecture Recovery using Concerns.

- Mo et al. [S21] show that by mapping components, connectors, interfaces and concerns, as well as their relations, into an extended augmented constraint network (EACN), it becomes possible to model all the defined architectural decay instances using EACN concepts, such as pairwise-dependency relations.

- Machine Learning models are used to understand if through the history of the existing architectural smells in the project it is possible to predict the presence of architectural smells in future versions in the framework by Fontana et al. [S22].

- Del Carpio [S23] describes a graph-based analysis technique for identifying ATD by non-uniformity of patterns.
- Martini and Bosch [S24] propose to use CAFFEA framework as a model to identify Social Debt. Which in turn will lead to the detection of ATD by analyzing the system for which the weak parts of the organization are responsible. Although this is not a direct implication, it can represent a first indicator of where ATD can be mostly accumulated. Since the analysis of ATD is costly to perform, identifying the parts of the system where such analysis is more probably needed would help prioritise resources.

4. Use Cases Definition and Evaluation

Contents

In this chapter, we define use cases of EA debt modelling based on the investigated works in this domain from the previous section. The description and definition part is followed by the evaluation to assess how good presented use cases display the need for debt modelling. The outcome of this chapter is the answer to RQ1.

4.1. Defining use cases

The next step is to formulate possible use cases of using modelling approaches proposed in analysed literature in the EA debt area. Presenting various situations would help to justify the need for EA debt modelling notation.

Use cases were created for each paper, using the template "As a – I want – so that." Even though there are multiple stakeholders who will potentially benefit from being able to model EA debt, an Enterprise Architect is a default actor for each story. This focus provides a possibility to cover the most fundamental needs which in a way extend to a number of other stakeholders as well. Initially, the possible use cases of using models in the context of EA debt and its management were formulated based on each paper from the review. But after the first analysis, two of them turned out to be not relevant. Thus, we had a list of 22 use cases grouped by their focus into 7 categories: Relevance, Strategy, Costs, Traceability, Detection, Dependency, and Compliance.

4.1.1. Relevance

Some modelling approaches are designed to be used as decision-making guidelines to communicate various aspects of debt and their relations to stakeholders. Such models

are needed to support enterprise architects, business analysts, managers, and business users from both a technical and a financial perspective (Table 4.1).

Use case ID	R1	R2	R3	R4
As an	Enterprise Architect	Enterprise Architect	Enterprise Architect	Enterprise Architect
I want/need to	have models and guidelines describing architectural use cases alternatives	have a unified model of EA debt	have models and guidelines describing the propagation of debt impacts and the vicious circles in the organisational models	have models and guidelines supporting topics of usability, efficiency, security, etc.
So that	I can compare possible debt impact and choose the best architecture to be implemented	I can present different aspects of debt and their relationships to the stakeholders	I can identify the presence of EA debt items and prioritise them	I make decisions on debt based on best practices

Table 4.1.: Use case definitions: group Relevance

4.1.2. Strategy

EA model could be used as a strong decision-support tool for analysing the properties of current and potential scenarios, planning and designing future scenarios and the path to get there. These modelling techniques focus on the motivations or rationale, i.e. the why, behind the architectures. With the support of such models, leadership together with architects would be able to evaluate decision alternatives and minimise the impact of debt on the organisation as a whole (Table 4.2).

4.1.3. Costs

There are models based on a cost-benefit analysis that provides the ability to analyse the debt related cost performance (e.g., measuring the amount of profit not earned due to the underutilisation of a given service and considering the probability of overutilisation of the selected service that would lead to accumulated debt). With the help of such models, the architect could predict the incurrence of the EA debt and the risk of entering into a new one in the future, as well as provide management with needed information to perform cost analysis (Table 4.3).

Use case ID	S5	S6	S7	S8
As an	Enterprise Architect	Enterprise Architect	Enterprise Architect	Enterprise Architect
I want/need to	perform an analysis of possible value-adding options (e.g., migration to the cloud)	locate and visualise the impact produced by the deliberately introduced EA debt	be able to model goals and requirements in EA	perform a security weakness analysis
So that	I can support changes and avoid possible debt	I can evaluate benefits of the possible repayment strategies over others decisions	possible debt situations can be detected in advance	I can identify the highest risks in an organisation with a potential to impact EA debt

Table 4.2.: Use case definitions: group Strategy

Use case ID	C9	C10	C11
As an	Enterprise Architect	Enterprise Architect	Enterprise Architect
I want/need to	perform cost-benefit analysis (measuring the financial consequences of the under-/over-utilisation of a given service)	analyse the debt related cost performance	have a template for EA debt items documentation that reports positive impact on the organisation when a debt item is incurred
So that	I can predict the occurrence of the accumulated EA debt	I can inform stakeholders and make weighted decisions	so that I can evaluate possible debt situations

Table 4.3.: Use case definitions: group Costs

4.1.4. Traceability

Adaptation to change is an important requirement for EA. In order to support the impact of change analysis, debt issues should be traceable to their roots and affected architecture elements; and vice versa. With this available, architects can quantify debt and quantify the expected pay-back, as well as be able to predict the occurrence of similar issues in the future (Table 4.4).

Use case ID	T12	T13	T14	T15
As an	Enterprise Architect	Enterprise Architect	Enterprise Architect	Enterprise Architect
I want/need to	be able to track debt architecture roots	be able to predict EA debt based on history data	be able to locate the architectural sources of EA debt	analyse the current organisation structure
So that	I can analyse and fix the cause, and avoid similar mistake in the future	I can predict the presents of the EA debt in the future	I can quantify them and quantify the expected pay-back for refactoring these debts	I can identify gaps in the architecture community and weaknesses of the organisation which could cause EA debt

Table 4.4.: Use case definitions: group Traceability

4.1.5. Detection

All debt causing issues that happen during the enterprise lifecycle can be classified. Moreover, with time it is possible to identify patterns of events that lead to such issues. Automatic locating of newly submitted issues will raise architectural awareness, help architects to prevent the adverse effects of EA debt, and help them to deliver better solutions based on well-informed decisions. (Table 4.5).

4.1.6. Dependency

It is important not only to spot an EA debt item but also to analyse its relations with other architectural elements and possible dependencies between them. Mapping debt elements to design decisions and their constraints will allow architects to model these architectural items as patterns, which, in turn, can be automatically and uniformly detected in the future (Table 4.6).

Use case ID	DT16	DT17	DT18
As an	Enterprise Architect	Enterprise Architect	Enterprise Architect
I want/need to	be able to identify patterns in EA that lead to debt occurrence	be able to identify the architectural significance of newly submitted issues	automatically detect the precise locations of EA debt
So that	I can analyse the causes and predict future issues	prevent the adverse effects of architectural decay	I can quantify the interest rate of each debt and to predict the cost of each debt in the future

Table 4.5.: Use case definitions: group Detection

Use case ID	DP19	DP20
As an	Enterprise Architect	Enterprise Architect
I want/need to	map EA debt to dependency models	identify dependency-related problems that are likely to appear in a system
So that	EA debt instances could be automatically detected in the future	I can prevent the issue from occurring

Table 4.6.: Use case definitions: group Dependency

4.1.7. Compliance

What constitutes a "good" enterprise architecture model is dependent on its purpose, i.e. the type of analysis it intends to support. For instance, in the case of analyzing cyber security, the property of whether it is possible to reconfigure a firewall is of interest. An effective EA model should be aligned in accordance with best practices so that the architect is able to capture ambiguities in system design (Table 4.7).

Use case ID	CM21	CM22
As an	Enterprise Architect	Enterprise Architect
I want/need to	perform enterprise architecture analysis	be sure that the enterprise coherence is explicit
So that	I can evaluate EA models with respect to best practices and standards, and build an effective architecture	stakeholders are satisfied and enterprise functioning as it should

Table 4.7.: Use case definitions: group Compliance

4.2. Use cases evaluation

The overall evaluation process can be described as follows. During the planning phase, we defined a goal, identified the target audience and assessed its feasibility, and determining methods for collecting data. The next step was to create the questionnaire and send it out to the domain experts and practitioners to gather feedback. The evaluation was one-way communication, meaning that the responses were analysed without involving participants or going back to them for additional information.

4.2.1. Design of the questionnaire

To evaluate how relevant each use case is to the EA debt topic, the questionnaire was created and distributed to field practitioners. The focus of this questionnaire is to rate how relevant each use case is on a scale from 1 to 5 (where 1 is not relevant and 5 is very relevant). The question answered by the questionnaire presented here is: "How relevant are the following use cases to the stakeholder?" The target audience of the questionnaire is EA experts and practitioners, the potential users of the new modelling notation.

The questionnaire is divided into 9 sections. The first section describes the purpose of the evaluation. Followed by sections for each of the use cases groups (namely, Relevance, Strategy, Costs, Traceability, Detection, Dependency, and Compliance). Each of these sections contains the rating of the corresponding use cases and a comment section asking for feedback (e.g., are there any use cases missing or ones that need improvement). The

section in the end contains the open-ended question to enter the feedback regarding the evaluation or the overall scope of this work.

4.2.2. Questionnaire results

This section presents the results of the evaluation grouped by the concept that is covered by the use cases. The questionnaire was sent out to field researches, students and people, who are involved in the decision processes on the product or senior management level in the organisation. In total, 9 opinions could be gathered through the use of Google Forms.

As an overall observation, almost every question has an option that got the majority of the votes. Only for a couple of questions, the majority of the practitioners' votes are distributed between two options. Regarding feedback, it can also be divided into two groups: comments that asked for the clarification of some terms or comments with ideas for possible improvements. Interestingly, there is one comment stating that "You really have to be an expert in EA debts to correctly respond to the questions", although we tried to omit highly specific terms and provided the literature references for things that might need clarification.

Following is the analysis of responses for each group of use cases. Complete data of the responses are included in Appendix B.

1. Relevance

   Use cases R1 (Figure 4.1) and R2 are rated as highly relevant (score 5), opposed to the use case R4 which got a neutral score 3 (somewhat relevant). Additionally, one comment states:

   "Perhaps the most important factor for debt is the inferred cost of not prioritising debt resolution, the final point has a fairly significant skew, for example, usability or efficiency are significantly less impactful than a security risk in a lot of circumstances."

   As for the use case R3, the votes are distributed between ratings 3 (somewhat relevant) and 4 (relevant). Based on that, it was decided to group this use case with use case R2 and formulate it as:

   "As an Enterprise Architect, I want/need to have a unified model of EA debt so that I can prioritise and present different aspects of debt (e.g., complexity, time perspective, maintenance, impacts and the vicious circles) and their relationships to the stakeholders."

   Besides, the definition of the unified EA model was not that clear to some of the practitioners. In the study by Besker et al. [BMB18], the unified model refers to a model that is used as a standard such that everyone (on the company scale or even in the EA field in general) is on the same page and operates the same vocabulary.

2. Strategy

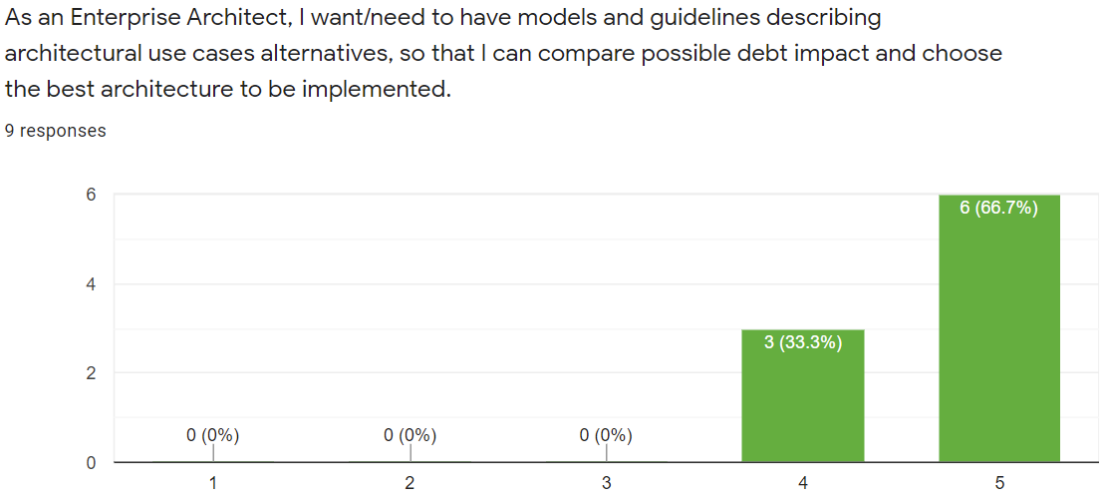

Figure 4.1.: Use case R1 evaluation: expert ranking

Moving to the Strategy category, every use case was rated as relevant (score 4) or highly relevant (score 5). One eye-catching example is the use case S5, which got 8 out of 9 votes for being very relevant (with one neutral rating of 3). Furthermore, there is a comment regarding use case S8:

"To the "identify the highest risks", I would add some regard for probability of the risk happening - if it is very low, we might not be interested in such risk. Also, I would briefly define what does "security weakness analysis" do."

Based on that, this use case is reformulated to:

"As an Enterprise Architect, I want/need to perform a security weakness analysis (e.g., using CWE or SWOT analysis) so that I can identify and prioritise risks in an organisation with a potential to impact EA debt."

3. Costs

Both use cases C9 and C10 were rated as highly relevant, as well as there were a couple of comments stating that they sound similar and might be either differentiated more or combined together. Thus, these use cases are grouped and reformulated to:

"As an Enterprise Architect, I want/need to perform cost-benefit analysis (measuring the financial consequences of the under-/over-utilisation of a given service) so that I can predict the occurrence of the accumulated EA debt, inform stakeholders, and make weighted decisions."

As for the use case C11, even though the majority of votes goes to the neutral rating 3 (4 votes), the sum of the answers for ratings 4 (relevant) and 5 (highly relevant) (1 and 3 respectively) will be identical. Thus, we are not excluding it from the list, but slightly modifying it based on the comments:

"I would report both positive and negative impact - what if the positive would not outweigh the negative?"

"What is the "template for EA debt items documentation" about? You should focus on the objective rather than the tool."

The final definition of the use case:

"As an Enterprise Architect, I want/need to have a template for EA debt items documentation that reports positive as well as negative impact on the organisation when a debt item is incurred so that I can evaluate possible debt situations."

4. Traceability

   Use cases from the Traceability group were mostly rated as being highly relevant, except the use case T13, where votes were distributed between ratings 2 (slightly relevant) to 5 (highly relevant). Additionally, there is a comment about use cases T12 and T13 being similar, so we decide to leave out the second one and choose the use case T12 as a representative for the idea of being able to track and analyse the cause of the debt item. Moreover, there is a comment regarding use case T15:

   "It is not clear how an organisational chart can be helpful in this case. Such an analysis should be about tasks and responsibilities."

   But because it was rated as being relevant to the group (relevant - 5 votes, very relevant - 3 votes), we leave it without changes.

5. Detection

   In the Detection section, all the use cases were rated as being relevant or highly relevant, so we leave everything as it is without further improvements.

6. Dependency

   Regarding the use case DP19, we decided to drop it due to the score (5 out of 9 practitioners graded it as being not relevant or neutral as shown at Figure 4.2) and comments ("robustness is more important than the method of detection"). The use case DP20 is graded as being highly relevant and did not receive any comments, so left without changes.

7. Compliance

   Both use cases from this group have almost identical grading (with the difference in only one vote). There is a comment regarding the use case CM22 to redefine "functioning as it should" part to something like "the enterprise functioning according to its guidelines". With that in mind, we change this use case to:

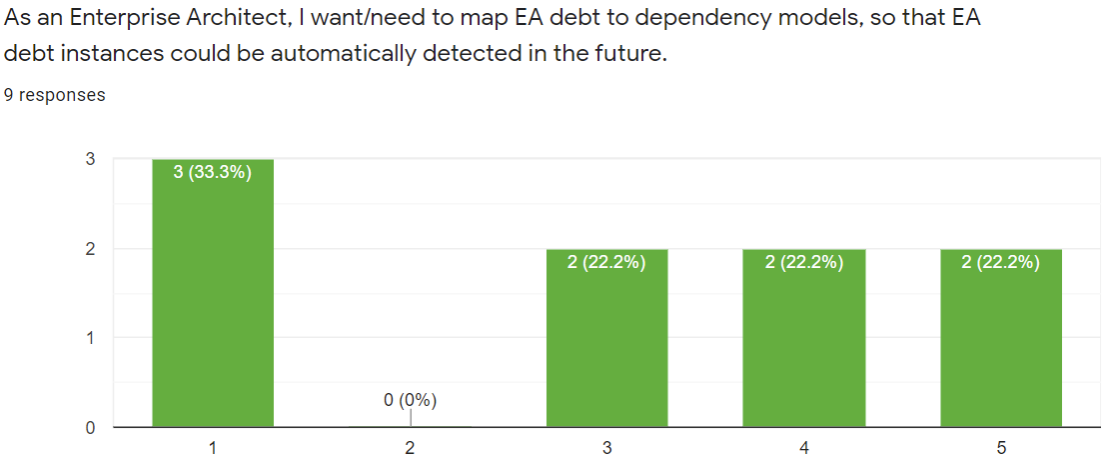

Figure 4.2.: Use case DP19 evaluation: expert ranking

> "As an Enterprise Architect, I want/need to be sure that the enterprise coherence is explicit so that stakeholders are satisfied and the enterprise functioning according to its policies".

Table 4.8 contains all the use cases that were changed during the evaluation. It shows the old and new version of the use case. In the case where use cases have been grouped together, the new new use case inherits the ID of the first parent.

Use case ID	Original definition	Updated definition
R2	As an Enterprise Architect, I want/need to have a unified model of EA debt, so that I can present different aspects of debt (e.g., complexity, time perspective, maintenance) and their relationships to the stakeholders.	As an Enterprise Architect, I want/need to have a unified model of EA debt, so that I can prioritise and present different aspects of debt (e.g., complexity, time perspective, maintenance, impacts and the vicious circles) and their relationships to the stakeholders.
R3	As an Enterprise Architect, I want/need to have models and guidelines describing the propagation of debt impacts and the vicious circles in the organisational models, so that I can identify the presence of EA debt items and prioritise them.	
S8	As an Enterprise Architect, I want/need to perform a security weakness analysis, so that I can identify the highest risks in an organisation with a potential to impact EA debt.	As an Enterprise Architect, I want/need to perform a security weakness analysis (e.g., using CWE or SWOT analysis), so that I can identify the highest risks in an organisation with a potential to impact EA debt.
C9	As an Enterprise Architect, I want/need to perform cost-benefit analysis (measuring the financial consequences of the under-/over-utilisation of a given service), so that I can predict the occurrence of the accumulated EA debt.	As an Enterprise Architect, I want/need to perform cost-benefit analysis (measuring the financial consequences of the under-/over-utilisation of a given service), so that I can predict the occurrence of the accumulated EA debt, inform stakeholders, and make weighted decisions.
C10	As an Enterprise Architect, I want/need to analyse the debt related cost performance, so that I can inform stakeholders and make weighted decisions.	
C11	As an Enterprise Architect, I want/need to have a template for EA debt items documentation that reports positive impact on the organisation when a debt item is incurred, so that I can evaluate possible debt situations.	As an Enterprise Architect, I want/need to have a template for EA debt items documentation that reports positive as well as negative impact on the organisation when a debt item is incurred, so that I can evaluate possible debt situations.
CM22	As an Enterprise Architect, I want/need to be sure that the enterprise coherence is explicit, so that stakeholders are satisfied and enterprise functioning as it should.	As an Enterprise Architect, I want/need to be sure that the enterprise coherence is explicit, so that stakeholders are satisfied and enterprise functioning according to its policies.

Table 4.8.: Updated use cases after evaluation

5. Results

Contents

This chapter describes the evaluation process of existing modelling frameworks to answer the question of how well they can fit the needs of EA debt modelling. After a brief overview, each modelling notation is used to visualise a given debt situation. We analyse created models for their use case coverage to see what elements are missing to provide a clear understanding of the situation. Based on this, we present a list of the main concepts required to model EA debt. The chapter ends with a description of the application of the developed notation in the EA debt management process.

5.1. Modelling notations evaluation

While well-established modelling methods are extensively employed in enterprise architecture, there is no work focusing on specifically EA debt modelling. To investigate to what extent existing modelling approaches can cover the debt topic, we will evaluate the most famous of them (i.e. those mentioned in section 2.2) following the next steps:

1. Define a scenario to be modelled.
2. Create a model using the chosen framework.
3. Perform an analysis of the resulting model.
    - Does the notation provide elements for visualizing all the items crucial for the understanding of the EA debt situation?
    - Is it possible to represent relationships between all architectural items?
    - Is it possible to replace debt-specific elements with already existing concepts?

5.1.1. Example situation

Hacks et al., in their study [Hac+19], come up with two examples showing the possible occurrence of EA debt. One of them is chosen to be used in this study as it can, with some extensions, cover the context of all defined use cases. It goes as follows.

"A company is situated in the insurance market and implemented its business critical applications on their mainframe until the end of the last decade. Due to a change in their IT-strategy, future applications should be developed using cloud environments. As the application landscape is comprised by more than 300 applications, a big bang scenario, where all applications are moved to the cloud, is unfeasible. Consequently, the applications are moved to the cloud step by step according to their application life-cycle rating.

The central enterprise architecture department has defined a target landscape and a road map describing the way to get to the target landscape. This road map includes also two applications a and b, which should be moved from the mainframe to the cloud. Both applications depend on each other, which means that they use interfaces of each other. As it is planned that both applications should be moved to the cloud simultaneously, the interfaces of both applications can be developed within the target landscape.

Due to unforeseeable delays in the project that implements application b, it is not expected that a and b can still go live at the same time in the cloud. However, the interfaces of b are indispensable for the use of a. Therefore, a is developed in a way that it relies on the interfaces of the mainframe implementation of b. As this interface implementation is obviously not part of the target landscape, this will introduce EA Debts into the EA.

Nonetheless, there is no feasible alternative and, therefore, there is the need to take this additional effort, which leads to a worse quality of the overall EA. However, the concept of EA Debt can help to create awareness for this quality issue along all stakeholders and that this EA Debt should be repaid as soon as application b has been moved to the cloud."

5.1.2. Models analysis

ArchiMate

The model on the Figure 5.1 demonstrates the visualisation of the example situation using the ArchiMate framework.

1. Even though, ArchiMate does not provide all concepts needed for debt modelling, there are some aspects covered by available notation. These are various architectural (Application Component, Application Interface, Node) and business elements (Goal, Business Process) needed to provide the context of the situation, Stakeholder (to see who has interests in the effects of the architecture). What is missing is the identification of the elements, causes and consequences of debt.

2. It is definitely possible to show that elements are related to each other, but available relationships types do not fully cover the scope of potential connections between

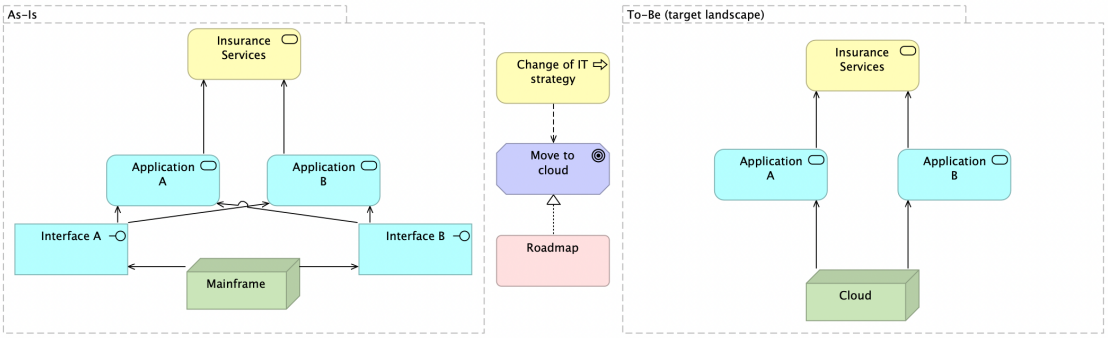

Figure 5.1.: Example of visualising a debt situation using the ArchiMate framework

objects in debt-related situations. Here it will be a time alignment between items (both applications should be moved to the cloud simultaneously). The functionality to show that processes are aligned with each other in time, have time dependencies or restrictions is needed to provide a clear understanding of possible time constraints in the model.

3. To some extent, it is possible to replace missing elements with already existing concepts, but not all of them. For the debt item, for example, one can simply put "debt" in the name of the element itself (and make it bold to stand out). And to represent processes that triggered changes (based on Traceability group use cases), one can use a grouping element. A possible solution for debt consequences is to represent them as business or technology processes. On the other hand, the separate unique element will make the model more clear to read.

BPMN

The model on the Figure 5.2 demonstrates the visualisation of the example situation using the BPMN framework.

1. To some extent, it is possible to cover cause (delays in the project that implements application b) and repayment (debt acceptance) concepts, but other debt specific aspect are missing (consequences and impacted parts of the architecture).

2. As BPMN is a process modelling approach with Activities and Events as the main elements in the notation, the relationships between elements represent their connection to each other in the flow. The main disadvantage here is that there is no meaningful way to model and show the connections between architecture elements. Available connections represent different types of flow, such as sequence flow, message flow or conditional flow. Thus, it is not possible to show the situation from the example that application A is dependent on the interface of application B.

3. In contradiction to ArchiMate, BPMN gives a better toolkit to represent the processes behind the current state of the architecture. So if there is a debt item,

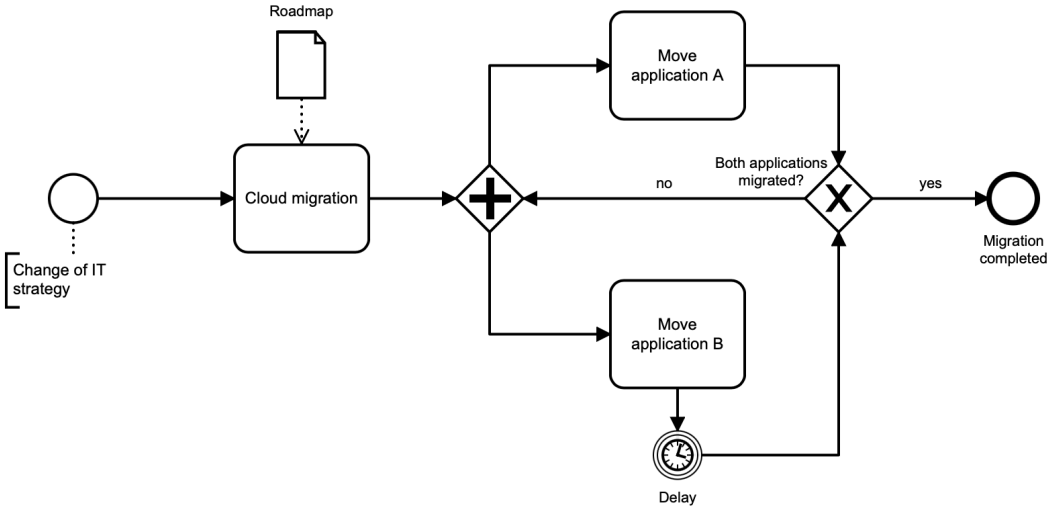

Figure 5.2.: Example of visualising a debt situation using the BPMN framework

one will understand what events led to it (which is required by the traceability use cases). But once again, there is no unique representation of the debt element. BPMN models can be useful when modelling repayment strategies (use case S2), as it is possible to add a roadmap element and visualise steps needed to resolve the issue. Another point is that Start and End Events, in a way, represent motivation and supposed outcome (goal). Additionally, a Timer Event can be used to represent the delay in the project mentioned in the example.

BMM

As BMM concepts can be mapped to ArchiMate Motivation Extension, we will not review it here as a separate model.

UML

The model on the Figure 5.3 demonstrates the visualisation of the example situation using the UML framework.

The basic UML elements are very similar to the ArchiMate notation, thus the models look alike. In this case, the points of why this notation does not cover the EA debt situations are also the same. One thing that is different here, is the presence of an Activity element, which can be connected to the "To-Be" diagram to represent the motivation behind the changes.

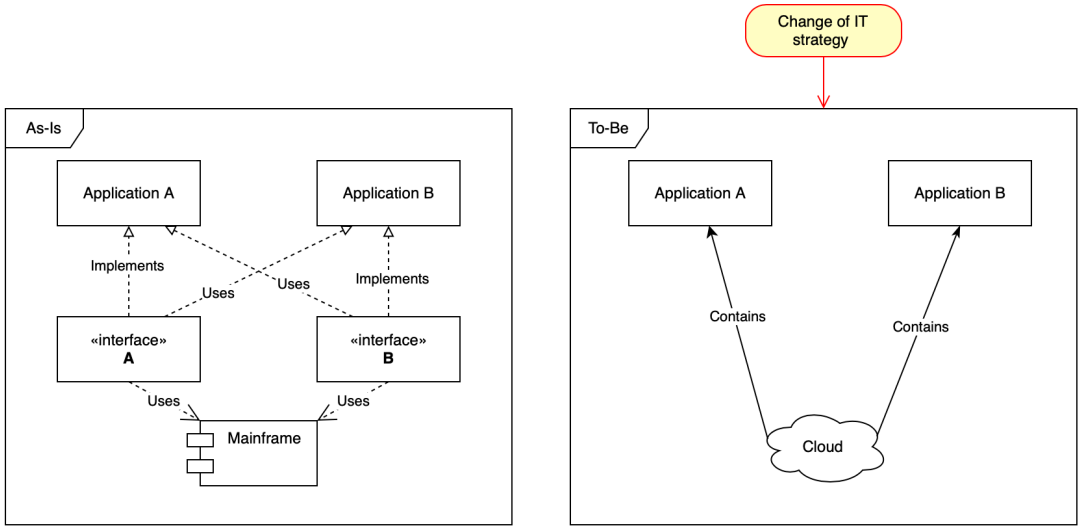

Figure 5.3.: Example of visualising a debt situation using the UML framework

5.1.3. Use cases analysis

The needs of use case R1 will be fully covered by creating different models for possible architecture alternatives, thus each model will represent each alternative. In the presented example, if the architect would have decided to analyse the possible ways of cloud migration, he would simply need to create multiple models. There is no need of creating new specific elements for this use case except previously mentioned ones from the analysis of other use cases. Similarly for the use cases from the Compliance group.

Talking about the unified EA model (use case R2), it is possible to create one that will be specialised to the needs of the organisation by using standardised guidelines as proposed in [BMB18], or use specifications of, for example, ArchiMate or any other modelling framework. The standardisation of the model throughout the company will ensure that everyone is following the same guidelines and operating the same vocabulary. Thus, every stakeholder will be able to track through such a model each aspect of interest.

As for the use case R3, we can argue that with existing frameworks for enterprise modelling there are already existing guidelines and best practices, although they would need to include the description of already mentioned missing elements to allow modelling of the situations where EA debt is present.

In a similar fashion, existing frameworks together with new elements will satisfy the point of the use case S5 to perform an analysis of possible value-adding options. An architect can create different models describing possible cloud migration scenarios to pick the most suitable one, as well as, support changes and avoid possible debt. The same conclusion also fits the use cases S8, C11, DT16, DT18, and DP20.

For use case S6, if the architect decides to deliberately introduce debt to the architecture to be able to analyse the possible consequences and risks, there is again a need to visually

highlight this element in the model to make it stand out. As there are no elements in analysed notations to present debt items, there is a need for a new one. Furthermore, it should also visually differ from the debt item element, to differentiate the situations of debt that is actually present in the architecture and possible or introduced debt. Similarly, newly submitted issues (use case DT17) also should be highlighted to provide clearance in the model.

The goals and requirements aspect (use case S7) is fully covered by the ArchiMate motivation elements (namely, BMM). As mentioned before, Start and End Event elements can represent this concept in BPMN. In UML this aspect is not covered by existing elements. Similarly, organisation structure (use case T15) is covered by business elements in ArchiMate and by Class Diagram in UML. While BPMN does not support it.

As for use case C10, to analyse the debt related cost performance one might need to understand what are the consequences (pain points or risks) of the present debt. There is no suitable element available in any notation, so there is a need to create such an element.

To be able to locate debt architecture roots (Traceability use cases), the trigger event should be displayed on a model. Such ArchiMate elements as Business/Technical Process (represents a sequence of business/technical behaviours that achieves a specific result), Business/Technical Event (represents an organisational/technology state change) could be suitable, but in order to be able to differentiate that the particular event (or event sequence) led to the occurrence of debt, there is a need to use a differently designed element. Particularly because the other two notations do not support that idea.

After performing the debt analysis and creating a repayment strategy, it is needed to inform the stakeholders. Creating a model will help to put everyone on the same page and easily explain the situation. In such a model there should be a possibility to display the type of measure. This information will also help to analyse the cost and benefit associated with an EA debt item and estimate them; to help with project planning and decision making. In ArchiMate, the Course of Action strategy element can be used for that purpose. In BPMN, this concept can be implemented as a separate model or a sub-routing of already existing flow. In UML, a /say Consequence tag can be added to the Object element.

5.2. Core concepts for modelling EA Debt

Based on the performed analysis, we can see that existed frameworks allow EA debt modelling only to some extend with some crucial concepts unavailable. We assume, that the most suitable notation for the EA debt modelling will be ArchiMate. It covers most of the needed aspects and is widely known in the field. Together with suggested additional elements, it will provide the functionality to model and analyse the EA debt issues. Following is a list of elements that should be designed to allow modelling of the EA debt situations.

1. EA debt item

An EA debt item represents instances or processes that depict a debt situation. This element can be further characterised by its prudence, presence, or intentionality which could influence a repayment strategy developed to resolve the resulting situation. The EA debt item has to be related to its cause and consequences so that stakeholders have a clear picture of which areas of the architecture are affected and can define the scope of work to overcome the debt.

2. Necessary/Unnecessary debt

   A necessary/unnecessary debt marks a debt item according to its prudence type. It distinguishes that the enterprise either has no other chance than to take the debts or that there is a possibility to invest directly in debt or to take the debt and maybe repay it later. This will help stakeholders prioritise and identify which EA debt items should be resolved first and which can be resolved later depending on the enterprise's business goals and preferences; can be useful for the analysis and planning of the future projects.

3. Possible/Existent debt

   A possible/existent debt marks a debt element according to its likelihood type. It differentiates which instances or processes might be an issue in the future or already are present in the architecture. It is useful for stakeholders in terms of identifying risks and predicting the occurrence of the EA debt.

4. Inadvertently introduced/Deliberately introduced debt

   An inadvertently introduced/deliberately introduced debt marks a debt element according to its intentionality type. It distinguishes a deliberately or inadvertently added debt element from the rest of the architecture. This information might be needed when developing a repayment strategy.

5. Cause

   A cause represents a process (internal or external) that led to changes in the architecture and triggered debt situation. These elements together with its relations to the EA debt item can be used for determining debt patterns for future projects.

6. Consequence

   A consequence represents an outcome of EA debt and its severity. Consequences are closely related to architectural elements or stakeholders that might be affected by the resulted situation. It has an important role in analysis of possible debt impact and affected parts of the architecture. Stakeholders can use this information for making new or changing existing architecture decisions in order to eliminate or mitigate the negative influences of an EA debt item.

7. Course of action

   A course of action is an approach or plan for configuring some capabilities and resources of the enterprise, undertaken to achieve a goal. A course of action represents what an enterprise has decided to do. Courses of action can be categorized as acceptance, mitigation, delay.

8. Stakeholder

   A stakeholder represents an individual, team, or organisation with interests in, or concerns relative to, the outcome of the architecture. A stakeholder typically associates a value to certain aspects of the enterprise (e.g., influences the measure applied to the debt item), and thus also its reflection in the enterprise's architecture. Examples of stakeholders are not only the board of directors, shareholders, customers, business and application architects, but also legislative authorities.

9. EA element

   A meta-element that represents possible parts of the EA (e.g., application, cloud service, business process etc.). This element visualises how and which architectural components can be affected by EA debt and its consequences. As EA debt item could be a part of the existent architecture, clearly it is dependent on other architectural items.

Table 5.1 represents which elements need to be used to satisfy the conditions of each use case. For example, to cover the needs of use case R1 from the relevance group, you need to use the concepts EA debt position, cause, effect, and element EA.

Use Case ID	EA debt item	Cause	Consequence	Course of action	Stakeholder	EA element
R1	x	x	x			x
R2	x	x	x	x	x	x
R4	x	x	x			x
S5	x	x	x			x
S6	x	x	x	x	x	x
S7	x	x				x
S8	x	x	x	x	x	x
C9	x	x	x			x
C11	x	x	x	x	x	x
T12	x	x				x
T14	x	x				x
T15	x	x	x			x
DT16	x	x				x
DT17	x	x	x			x
DT18	x	x				x
DP20	x	x	x			x
CM21	x	x	x		x	x
CM22	x	x	x		x	x

Table 5.1.: Concepts used to support a use case

5.2.1. Meta-model

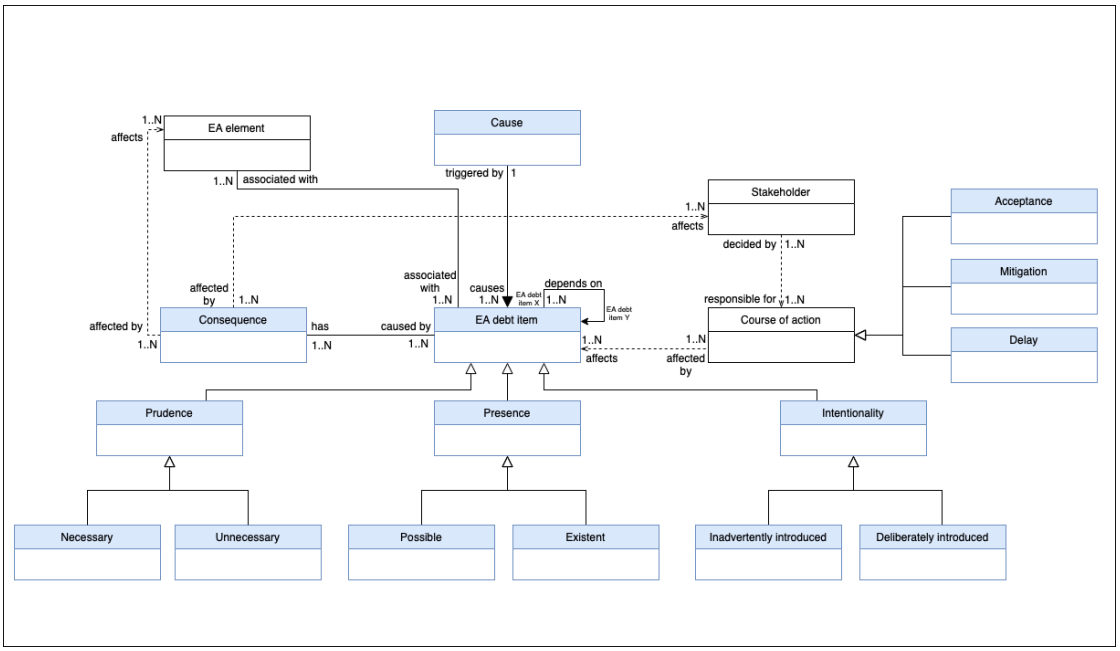

Figure 5.4.: Meta-model of core concepts and their relations

Figure 5.4 depicts the meta-model of the proposed extension. The coloured classes represent the newly introduced concepts, the white classes represent already existing concepts from other modelling notations.

Concepts from the debt domain are: EA debt item, Cause, Consequence, Course of action. The relations between these elements are based upon the existing relations from ArchiMate: Association, Trigger, Influence. The debt item is further specialised by following factors:

1. Prudence (Necessary and Unnecessary debt item);
2. Presence (Possible and Existent debt);
3. Intentionality (Inadvertently and Deliberately introduced debt item).

The Course of action is specialised in:

1. Acceptance,
2. Mitigation,
3. Delay.

The EA debt item and Course of action decomposition relation is a specialisation of the ArchiMate aggregation relation.

The idea is to use presented elements in combination with any other modelling framework. Thus, such concepts as EA element and Stakeholder are used here as meta concepts to represent the general idea behind the element which might be further specified by framework-specific notation. Both elements can be connected with the Consequence element using Influence relation, as it is clear that such elements of the architecture can be affected by the debt. Additionally, the EA element can be related to the EA debt item as these elements can be dependent on each other. Moreover, the Stakeholder element can be linked to the Course of action element using Influence relation, as the decisions on how to proceed come from people who are involved in the process or affected by debt.

5.3. Demonstration of using the modelling approach

Figure 5.5 shows how defined elements can be used in modelling debt-related situations. Here, the ArchiMate framework is used to show one more possible visualisation of the previously mentioned scenario (refer to section 5.1.1). The white instances on the model represent the newly introduced concepts, namely, Cause and Consequences. Grouping notation depicts elements that are classified as debt items in the architecture. Elements with names of concepts in brackets denote core concepts needed to model EA debt situations defined in this work.

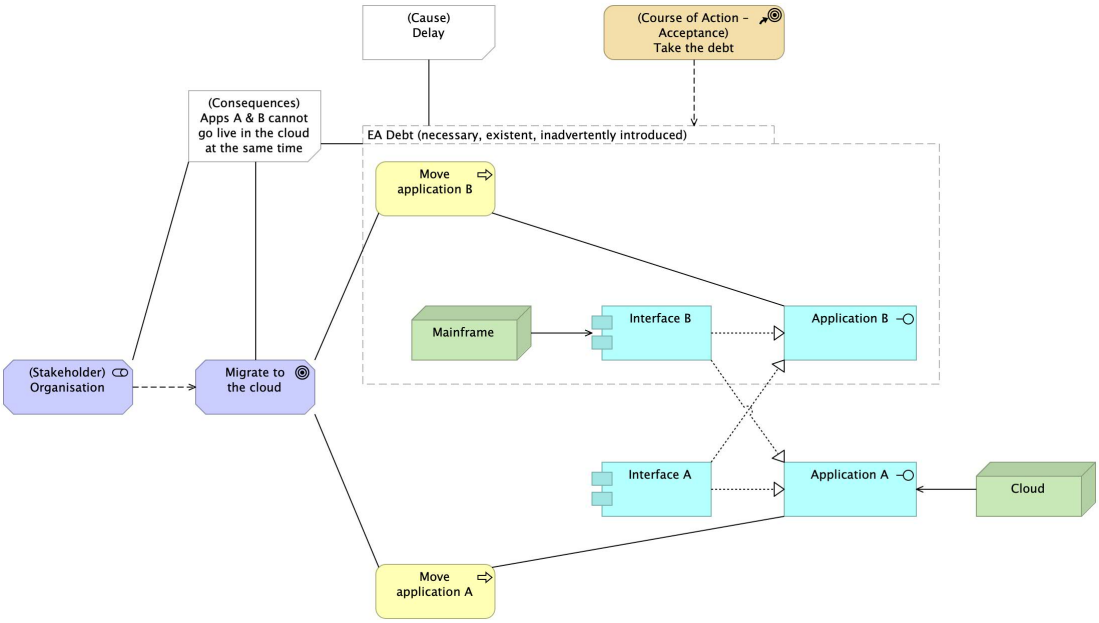

Figure 5.5.: Demonstration of the developed approach to modelling debt situations using the ArchiMate framework

This model now contains all the initial points of interest for stakeholders. It allows to

get familiar with the debt situation, understand what was the cause, which architectural parts are affected, and what are the consequences. A model like this will help connect the dots and ease the communication between all interested parties providing the same level of input for everyone.

6. Evaluation

Contents

In this chapter, the strategy of the evaluation of the defined debt modelling concepts and designed meta-model is described, followed by the summary of conducted interviews with filed practitioners.

6.1. Interview design

To evaluate the designed notation and its meta-model, we conducted interviews with field practitioners. The purpose of the evaluation is to assess such points of interest as task-oriented usability of the notation, its effectiveness, efficiency, and general design quality.

Steps of the interview session are:

1. Brief topic introduction.
2. Explanation of the notation elements and its meta-model.
3. Presentation of the situation to be modelled.
4. Performing the modelling task.
5. Answering follow-up questions.
   a) How well the meta-model represents the core concepts of debt?
   b) How intuitive the notation feels (i.e., without reading the definitions of the elements)?
   c) Does the notation allow to model debt-related situations?
   d) Can the model be used to clearly communicate debt situations to other stakeholders?

A total number of six people took part in the interview. A student with a business informatics background, two people from the EA debt research community, and three people working in a company who are directly involved in the decision processes on the product or organisation level. The average time of the evaluation was 35 minutes. One

interview took place in person and five online via Zoom. The practitioners were given the meta-model, the list of notation elements and their definitions together with the situation that shows EA debt as a PDF file. The main task is to draw a model of the debt situation using the given notation. During the task, practitioners were allowed to check with provided concepts definitions. Afterwards, they were asked previously mentioned questions (and follow-up questions if needed) to ask for possible improvement and analyse their satisfaction with the process.

6.2. Evaluation summary

Overall, the feedback from all practitioners was positive. Everyone mentioned that results so far look satisfactory, and with some improvements, the notation promises to be a practical addition for the EA domain.

How well the meta-model represents the core concepts of debt?

Answering the first question, all practitioners agreed that the meta-model is easily understandable, captures the main concepts, and element labelling provides an initial understanding of the meaning behind it. It has been mentioned that providing a meta-element that hides behind all possible architectural concepts (i.e., EA element) is a good idea, but two concepts that are still worth being presented separately are goals and requirements since they are closely related to some architectural elements and might be strongly affected by the consequences of the debt. The suggestion was to include them as separate items on the meta-model to indicate their importance.

How intuitive the notation feels?

As for the model intuitiveness, a student noted that "if you have enough knowledge about EA debt, this model will be easy to follow". Also, they mentioned that they needed to refer to the concept definitions a couple of times to check if they understood right the categories of EA debt item or the idea of EA element. Field experts said that wording is good, except maybe a Prudence category, as this term is quite domain-specific (two people needed to re-read the definition of this notion). These observations bring the point that even if mentioned concepts are commonly known in the context of technical debt and EA, people who are not that closely related to the domain could struggle to understand them.

Does the notation allow to model debt-related situations?

Answers to the third question brought the most points to refine for the model improvement. The main directions mentioned are adding cost or longevity attributes to some elements. Without generalising, here we will present the responses given by each interviewee.

1. The practitioner suggested adding attributes to the elements to give the model more context, for example, the price tag for the Mitigation.

2. “I would like to have cost tags for consequence attributes. Displaying it on the model will provide more information to start with the analysis.”

3. A recommendation is to add some type of scoring mechanism for risk analysis, for example, debt or consequence severity levels.

4. A proposal to add longevity attribute to the EA debt item (e.g., duration in weeks or months, or an S-M-L-like size) to indicate which kind of situation (short or long-term) the organisation will be dealing with. Another suggestion is to add a cost attribute that will show the price for having the issue fixed or kept in the architecture.

5. An expert suggested using enumeration instead of decomposition for EA debt item properties. They stated, that enumeration will allow extension in cases where the categories turn out to be non-binary.

6. “As for me, using this notation is a quick way to represent the debt situation.”

Can the model be used to clearly communicate debt situations to other stakeholders?

As for the fourth question, practitioners were saying the model is not complicated and overwhelmed with different concepts; all core information is included in the model to get familiar with the debt situation and take steps to define repayment strategy. But, people with management backgrounds remarked that the visual of the model might change depending on the viewpoint. For example, it might include more technical elements for the engineering perspective but be more linear and present only cause-debt item-consequences flow to higher management. Another comment was regarding the use of the models in the organisation in general. If visual representation is not a common way of transmitting the information, getting used to such a tool might take a while before everyone will feel comfortable and begin using models as a day-to-day practice.

Additional comments

There was one more comment about the general look of the meta-model. Namely, to get away from using the language-specific relations between elements (i.e., Trigger and Influence relations from ArchiMate) and use only Association relation. “Having both specific relations and word labelling is a bit overwhelming. Using Association will clean up the view of the meta-model.”

Two practitioners who previously took part in the use case evaluation were asked one additional question: "How well the notation covers the scope of defined use cases?". Both agreed that the visual model itself cannot solve all the needs of an enterprise architect and involved stakeholders, but it brings value as initial information input to such processes mentioned in the use cases as costs analysis, predicting debt patterns, building dept prevention strategies, or EA evaluation. On the other hand, it serves

greatly for identifying debt elements and their architectural roots, as well as for analysing the impact of debt on an organization.

7. Discussion

Contents

In this chapter, the discussion is divided into three parts: the analysis of the benefits, challenges and suggested improvements, the addressing of the research questions, and analysis of the possible threats to this work validity.

7.1. Benefits, challenges and improvements

The main outcome of the evaluation process is not the models created by interviewees but the domain experts' thoughts and reflections upon the process. Even though the chosen situation is taken from real-world practice (a couple of practitioners mentioned that they had to deal with something very similar), the evaluation environment was still artificial. The modelling will be applied out in the industry and used for defining debt situations for an organisation, and the people participating will be not only enterprise architects but also other stakeholders interested in the outcome of the architecture. The field experts who participated in both use cases and meta-model evaluations could therefore not reflect on the specific needs for their specific systems, but their comments and thoughts of the process can be seen as implications of how well and if the modelling approach would be possible to use. The benefits and challenges of the application of the EA debt modelling in the organisation, as well as mentioned possible improvement suggestions, are the main input points for future research.

The main benefit of using visual models for representing the EA debt situation is that it provides a unified way of communicating the information. According to Banaeianjahromi and Smolander [BS19], lack of communication and collaboration when it comes to EA is common and something that needs to be addressed. Also, that this approach could be used by coworkers for brainstorming together can be seen as an advantage. Since every concept has its purpose and can be used in one defined way, it limits the room for misinterpretation and false assumptions. It is important to everyone in the company, from an engineer to an external stakeholder, to use the same grammar, as it will align people in the process of achieving goals. On the other hand, it is time-consuming, and

therefore it has to be well motivated for companies to be able to invest time in conducting it. Since EA debt is not yet an established concept in the industry, the adoption and means of motivation regarding EA debt modelling, and hence a process for defining such, depends on when this will happen. The area of EA debt is still very novel, and this can also be argued to be seen in the results.

During the meetings, visualising the debt situation will help to focus on where exactly a pain point is located in the architecture, what parts of the architecture are affected and what action steps will be needed. Moreover, it will be possible to quickly go back to the created model and analyse how the situation has changed.

The documentation and communication activities of the EADM ensure that the flow of EA debt knowledge among the concerned stakeholders is maintained during the whole process. The key point in carrying out these tasks is to present and structure the complex EA debt information in an intuitive and user-friendly manner. Here the use of a visualisation tool will provide clearance and information sustainability as it is easier to refer to a model than to a passage of text. Thus, models play an important part in every decision-making process throughout the organisations. Enterprise architects use models to present possible architecture alternatives; managers use them to provide analysis of costs or risks to the stakeholders. Therefore, any framework for managing EA debts will not be complete without a debt visualising toolkit. Taking into account that there are a number of modelling frameworks used in different organisations, the debt-related notation has to be easily adjustable to fit the main concept. This is something we tried to do when designing the notation presented in this work. Introducing meta-elements that represent concepts from other modelling frameworks and reusing already available elements will allow practitioners to use new elements without changing the approach they already working with.

The ability to clearly distinguish architectural debt items and their connections to other elements in the model will improve the overall understanding of the situation without going deep into the documentation. Of course, good models do not completely remove the requirement of writing documents but they are very useful additions and even serve as the main input to such tasks as analysing possible alternatives, defining dependencies, or identifying risks.

One of the foreseeable challenges mentioned by a couple of experts is that not every organisation (or team inside an organisation, if talking about big scale companies) are familiar with using models as a communication tool. Yes, for the engineer or enterprise architect, it is common to represent their ideas as models. But for the higher management, who mostly talk about decisions and costs, creating models to spread the awareness of a specific situation might feel counter-intuitive and even a waste of time. But, as mentioned before, using models for communication will provide a unified knowledge ground. Thus, a company initiative to hold internal workshops will encourage people to apply modelling more actively in their work.

One of the possible improvement points that were mentioned during the evaluation is to use enumeration instead of inheritance to represent different characteristics of EA debt items. The justification point was that it will allow easier expansion of the list of possible

traits that the debt element can have. But on the other hand, differentiating these traits into corresponding categories makes it more intuitive to understand which concepts are being applied. And it is not assumed that these categorisations are binary. By conducting more empirical studies and finding more scenarios where debt modelling can be implemented, it might be proven that presenting dent characteristics as enumeration is indeed more meaningful.

Another point that we would like to address here is the suggestion to introduce attributes for some elements, which will add more context to the model. Based on the comments provided in the section, the following is the list of attributes that will be additionally included in the meta-model.

1. Costs for the Course of action and Consequence elements.

   Cost characteristics reflect expenditures on debt. These may be the cost of having debt in the architecture, the total cost of resolving the issue, the cost of the repayment strategy. Such price tags will be used in cost analysis and should be communicated to stakeholders (especially top management).

2. Severity level for the Consequence element.

   Severity marks how badly debt consequences will affect architectural elements they are related to. Based on the level assigned, stakeholders can prioritise what aspects of debt to take care of first or what repayment strategy will be the best option. It can be estimated with the complexity scale normally used in the company (e.g., Fibonacci numbers, T-shirt sizing, etc.).

3. Longevity for the EA debt item element.

   Longevity characteristic represents an estimated time debt will be present in the architecture. This information will be needed during planning and developing a repayment strategy. This attribute can be displayed in a number of weeks or using the T-shirt sizing estimation technique.

Applying one more recommendation to use only Association relation in the meta-model, figure 7.1 displays its final version.

7.2. Answering research questions

The first research question together with its sub-questions is addressed by conducting the SLR and defining the debt modelling use cases. Unfortunately, due to the novelty of the EA debt topic, there was barely any research available on the subject of analysing EA debt modelling. That is why it was important to look into the neighbouring topic of technical debt modelling. Based on this information, we were able to reuse some of the modellings requirements and determine the use cases for specifically EA debt modelling.

These use cases served as a ground floor for defining core elements needed for visualising debt-related situations. The list of such elements presented in chapter 5 together with the additions presented in the previous section, answers the second research question.

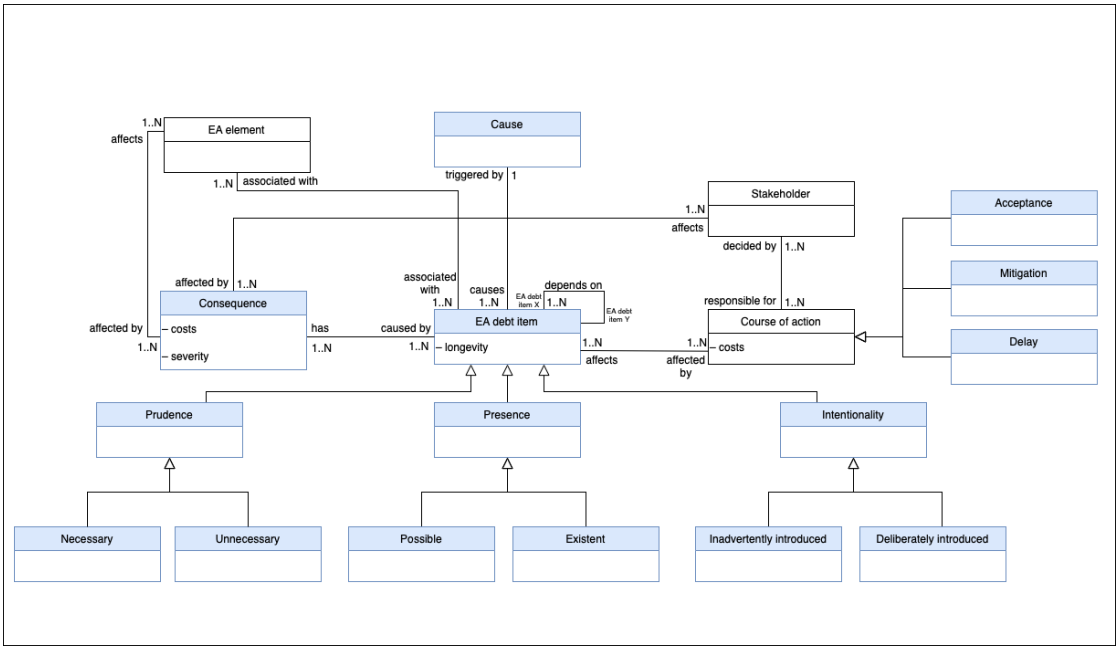

Figure 7.1.: Improved meta-model of core concepts, including their attributes and relations between elements

Thus, the actual definitions of these items together with the meta-model that represents the possible relations between all elements is the input to answer the third research question. But as it refers to the modelling notation, which includes not only the concept definitions but visual elements as well, this is something to pick up in future research.

7.3. Threats to the validity

7.3.1. Internal validity

Again, the novelty of EA debt has been the root of both the delimitations and the limitations of the results during this study. The debt topic is not yet broadly applied in the EA domain, and there are not many studies and experts available. As EA debt awareness becomes more widespread, EA debt modelling will become more relevant, which in turn will facilitate the study of different approaches.

Another threat to validity can be seen in the missing part of the quality assessment in the SLR. But our comment on why this is not a major drawback is that the number of studies available is quite low, to begin with. If certain quality criteria are to be applied, it could potentially shorten the final list of studies. Which, in turn, will affect the number of defined debt modelling use cases. Even if some studies were not entirely relevant to this work, use cases based on them were filtered out during evaluation.

The number of received questionnaire responses and interviews conducted also affect

the results. Indeed, all the collected feedback was valuable, and it was possible to find some patterns for improvement and link them to the needs of having such a modelling approach. But to fully ensure saturation when it comes to feedback sessions, more would have to be conducted.

7.3.2. External validity

This study highlights some of the use cases for debt modelling, but they might also not cover all the needs of some particular organisation, as each company has different goals and objectives. Only EA experts were interviewed and participated in the use cases evaluation, and they might have different drivers and objectives than other stakeholders. This again tangents the business-IT alignment that EA debt and its modelling can have an active part in overcoming. Another threat to the external validity comes from the fact that we did not select interviewees randomly, but contacted certain experts directly. As a result, our findings are limited in how they can be applied to different types of architects at different companies. Further research is needed to reflect more uses of debt modelling in EA and overcome expert bias.

8. Conclusion

Contents

To conclude this thesis, the key contributions of this work are summarised first, followed by several ideas that can serve as entry points for future research.

8.1. Summary

In this thesis, the idea of using a modelling approach has been introduced to provide a toolkit for the communication and documentation of the possible EA debt situations in the organisation. The modelling notation has been designed to serve as an extension for different EA modelling frameworks. It improves the decision-making processes by allowing quick identification and analysis of the debt items. It also introduces structure through the visualisation of dependencies between architectural elements and debt items. Having EA models which include debt aspects in them will increase awareness of stakeholders as well as ease processes of discovering deficits in the design of the EA.

The contributions of this work begin with the SLR of the research in the field of EA, EA debt, and the related areas of technical and architectural technical debt. The retrieved work was analysed in correspondence with the viewpoints supported by each model. As a result of this analysis, use cases for using debt modelling in an organisation by an enterprise architect were identified. They were evaluated by field practitioners and the final list of 19 use cases was formed. With the help of three modelling frameworks (ArchiMate, BPMN, UML), we have shown to what extent they can cover the scope of defined use cases. It resulted in the list of core concepts needed to be able to model EA debt situations. The meta-model of these concepts and their relationships has been evaluated by experts and, with several improvements, has proven to be a good ground input for improving the EA debt domain. Last but not least, the benefits and challenges of the presented modelling approach were discussed, as well as threats to the validity of this work.

8.2. Future Work

The main direction for future work will be designing visual notation to represent debt modelling concepts presented in this work. Those interested in UI/UX field can move on

to creating ArchiMate-like icons to match the defined definitions. It is crucial to create graphical elements that will be self-explanatory, easy to grasp and interpret. Designing intuitive visual components which support model customisation will give experts the opportunity to tailor the graphical representation of their models without changing debt modelling concepts. Additionally, to spread modelling as a communication tool throughout the organisation, it might be useful to create user-friendly guides that will help people to familiarise themselves with the usage and creation of visual models.

Afterwards, the next step will be to formalise the designed notation and integrate it into expert practices in real organisation contexts. It will allow investigating the correlation with architectural principles, business rules, and the elaboration of various analysis techniques, not only by using existing studies referred to in this work but also by exploring real-life situations. The propagation of the use of the developed notation will improve the overall EA debt managing experience. By triggering critical points, it will prompt researchers and practitioners to get insights on how to further develop ways of the organisation adopting EA and EA debt.

One more direction for future research is to look into more existing modelling frameworks to find further applicable concepts. It is worth exploring other related areas, such as software engineering, to define more modelling use cases and analyse if they can integrate into the EA debt domain.

A. Selected studies in the SLR

S01 M. Buschle et al. “A tool for enterprise architecture analysis using the PRM formalism”. In: International Conference on Advanced Information Systems Engineering. Springer. 2010, pp. 108–121.

S02 W. Engelsman et al. “Extending enterprise architecture modelling with business goals and requirements”. In: Enterprise information systems 5.1 (2011), pp. 9–36.

S03 C. Izurieta et al. "Preemptive Management of Model Driven Technical Debt for Improving Software Quality". 2015, p. 31.

S04 Y. Guo and C. Seaman. “A portfolio approach to technical debt management”. In: Proceedings of the 2nd Workshop on Managing Technical Debt. 2011, pp. 31–34.

S05 R. Wagter, H. E. Proper, and D. Witte. “A practice-based framework for enterprise coherence”. In: Working Conference on Practice-Driven Research on Enterprise Transformation. Springer. 2012, pp. 77–95.

S06 K. Ramchand, C. M.B., and R. Kowalczyk. “Towards a flexible cloud architectural decision framework for diverse application architectures”. In: ed. by K. Ramchand, M. B. Chhetri, and R. Kowalczyk.

S07 E. Alzaghoul, R. Bahsoon, and IEEE. "CloudMTD: Using Real Options to Manage Technical Debt in Cloud-Based Service Selection". 2013, p. 55.

S08 G. Skourletopoulos et al. “A fluctuation-based modelling approach to quantification of the technical debt on mobile cloud-based service level”. In: ed. by G. Skourletopoulos et al.

S09 Y. F. Cai, R Kazman, and IEEE. "Software Architecture Health Monitor". 2016, p. 18.

S10 L. Xiao et al. “Detecting the Locations and Predicting the Costs of Compound Architectural Debts”. In: IEEE Transactions on Software Engineering (2021).

S11 L. Xiao et al. "Identifying and Quantifying Architectural Debt". 2016, pp. 488–498.

S12 R. Kazman et al. "A Case Study in Locating the Architectural Roots of Technical Debt". 2015, p. 179.

S13 C. Izurieta et al. "A Position Study to Investigate Technical Debt Associated with Security Weaknesses". 2018, p. 138.

S14 B. Perez et al. "A Proposed Model-driven Approach to Manage Architectural Technical Debt Life Cycle". 2019, p. 73.

S15 T. Besker, A. Martini, and J. Bosch. “Managing architectural technical debt: A unified model and systematic literature review”. In: Journal of Systems and Software 135 (2018), pp. 1–16.

S16 Z. Li, P. Avgeriou, and P. Liang. “Architectural Debt Management in Value-Oriented Architecting”. In: Economics-Driven Software Architecture (2013).

S17 J. Saat et al. “Enterprise architecture meta models for IT/business alignment situations”. In: 2010 14th IEEE International Enterprise Distributed Object Computing Conference. IEEE. 2010, pp. 14–23.

S18 A. Martini and J. Bosch. "The Danger of Architectural Technical Debt: Contagious Debt and Vicious Circles". Ed. by L. Bass, P. Lago, and P. Kruchten. 2015, p. 1.

S19 A. Tommasel and IEEE. "Applying Social Network Analysis Techniques to Architectural Smell Prediction". 2019, p. 254.

S20 A. Shahbazian et al. "Toward Predicting Architectural Significance of Implementation Issues". 2018, pp. 215–219.

S21 R. Mo et al. "Mapping Architectural Decay Instances to Dependency Models". 2013, p. 39.

S22 F. A. Fontana et al. "A Study on Architectural Smells Prediction". Ed. by M. Staron, R. Capilla, and A. Skavhaug. 2019, pp. 333–337.

S23 P. M. D. Carpio. “Using naming patterns for identifying architectural technical debt”. In: Advances in Science, Technology and Engineering Systems 2 (2017), pp. 248–254.

S24 A. Martini, J. Bosch, and IEEE. "Revealing Social Debt with the CAFFEA Framework: an Antidote to Architectural Debt". 2017, p. 179.

B. Questionnaire results

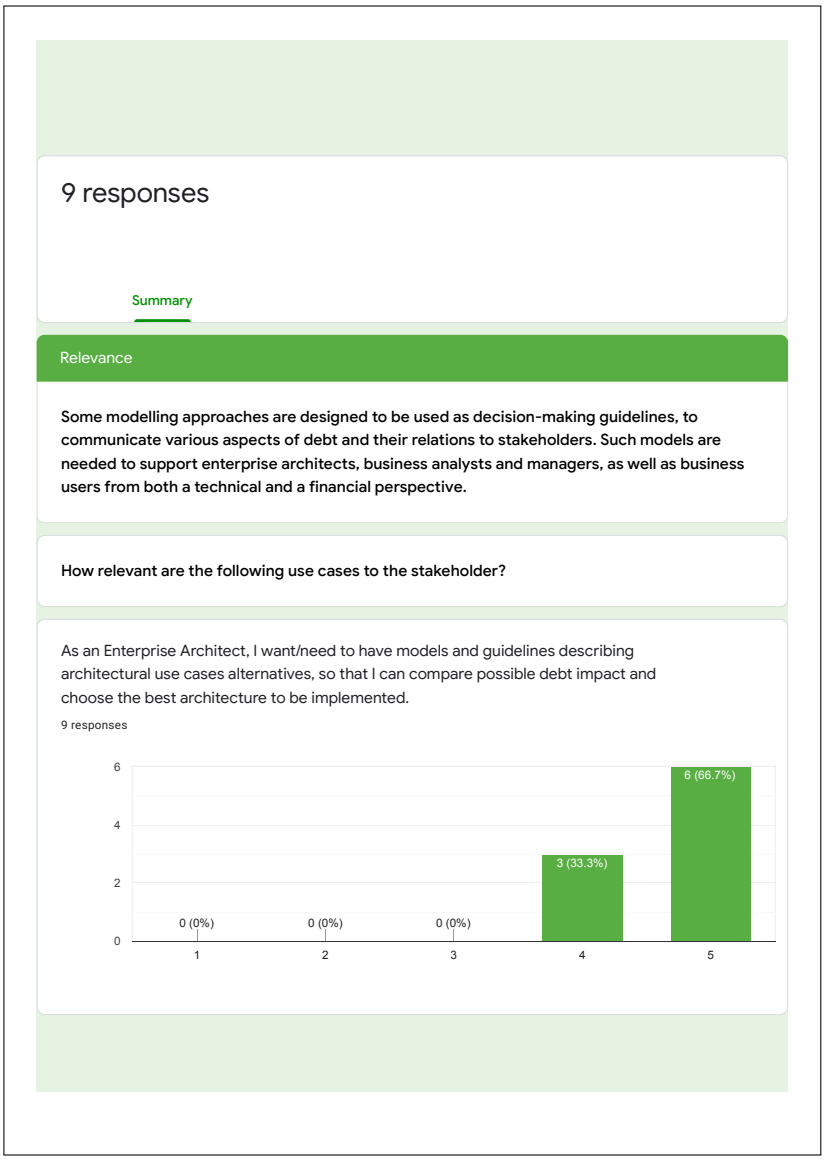

Figure B.1.: Questionnaire results page 1 of 15

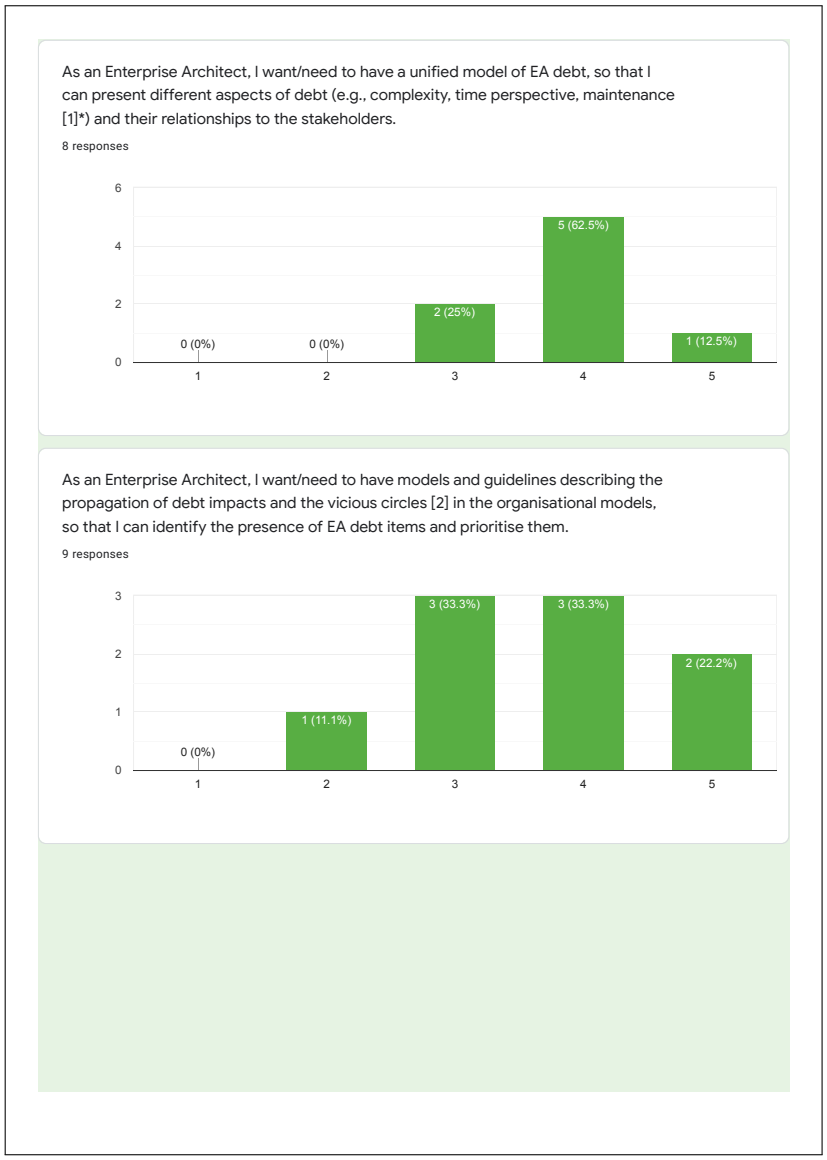

Figure B.2.: Questionnaire results page 2 of 15

Figure B.3.: Questionnaire results page 3 of 15

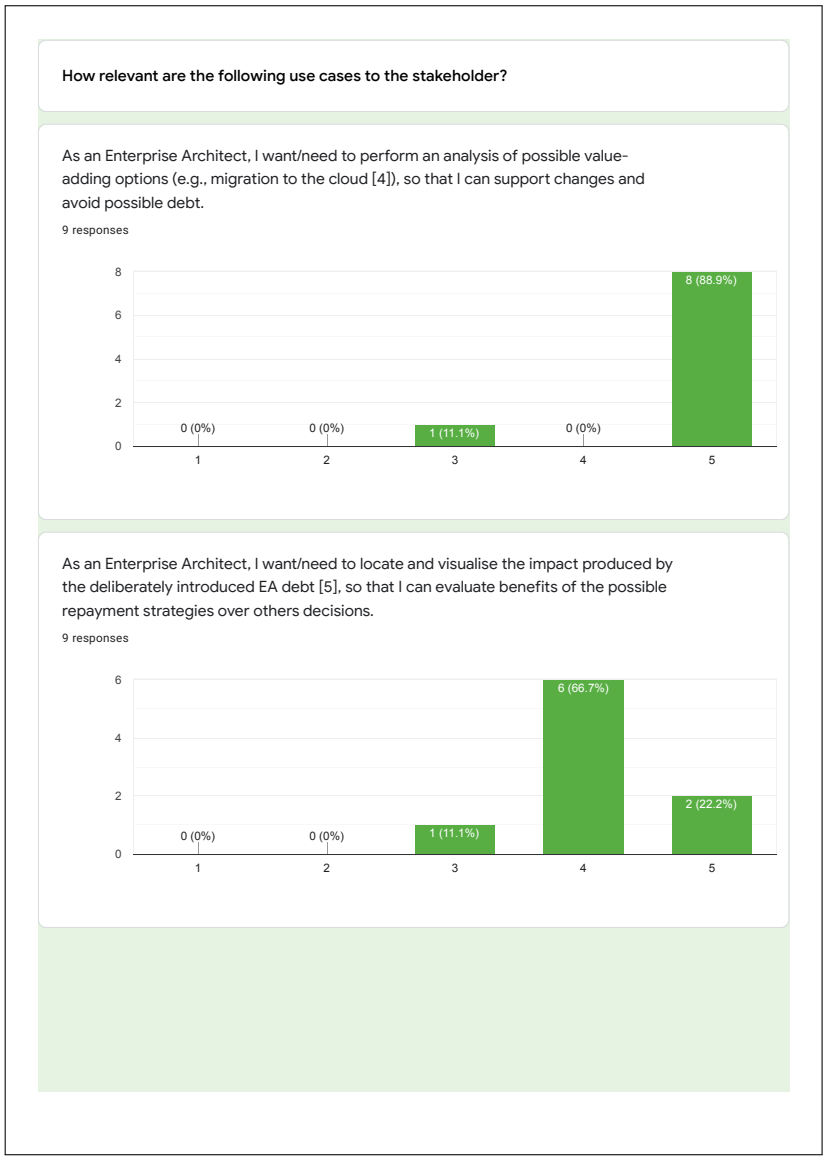

Figure B.4.: Questionnaire results page 4 of 15

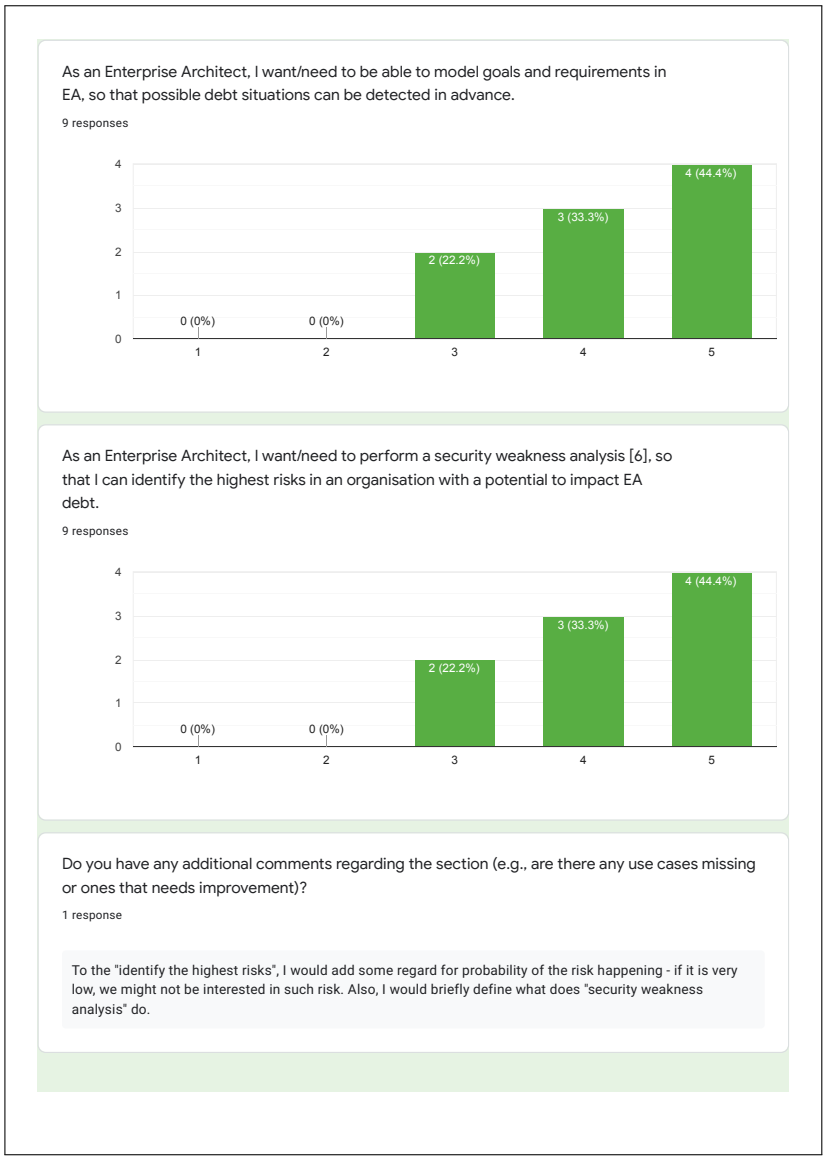

Figure B.5.: Questionnaire results page 5 of 15

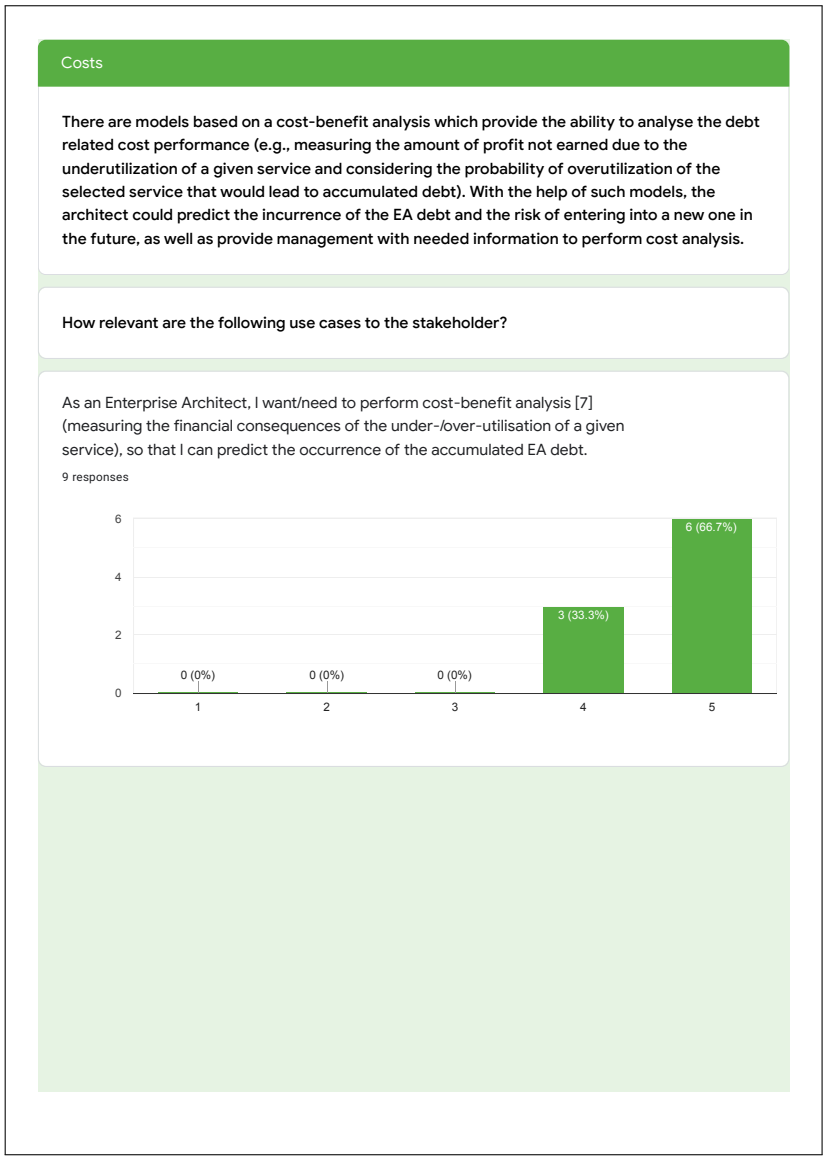

Figure B.6.: Questionnaire results page 6 of 15

Figure B.7.: Questionnaire results page 7 of 15

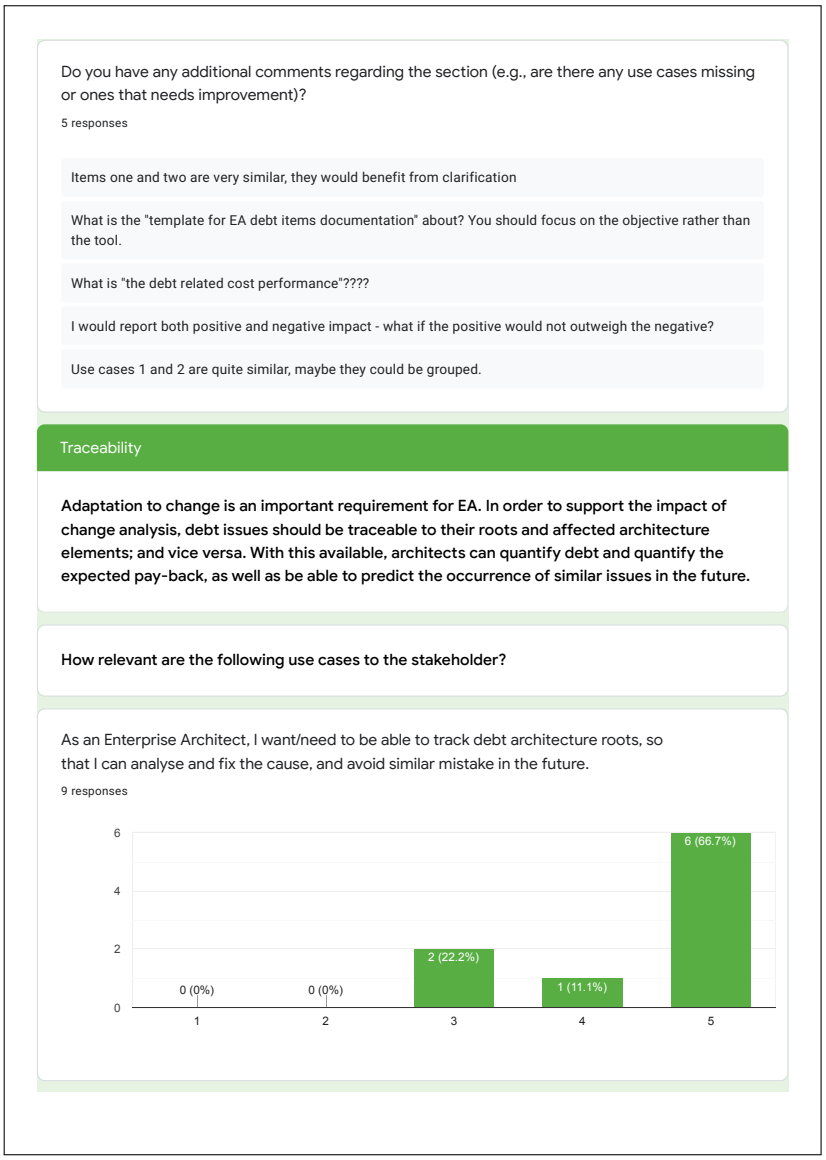

Figure B.8.: Questionnaire results page 8 of 15

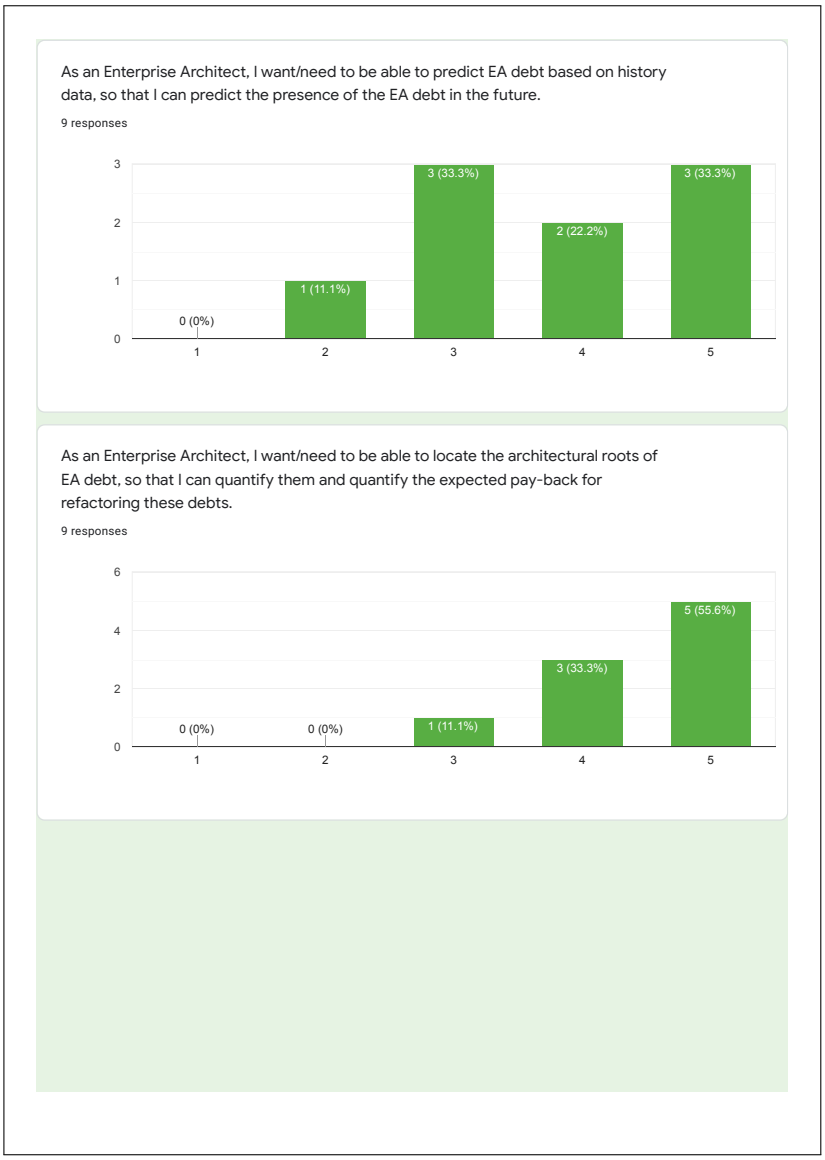

Figure B.9.: Questionnaire results page 9 of 15

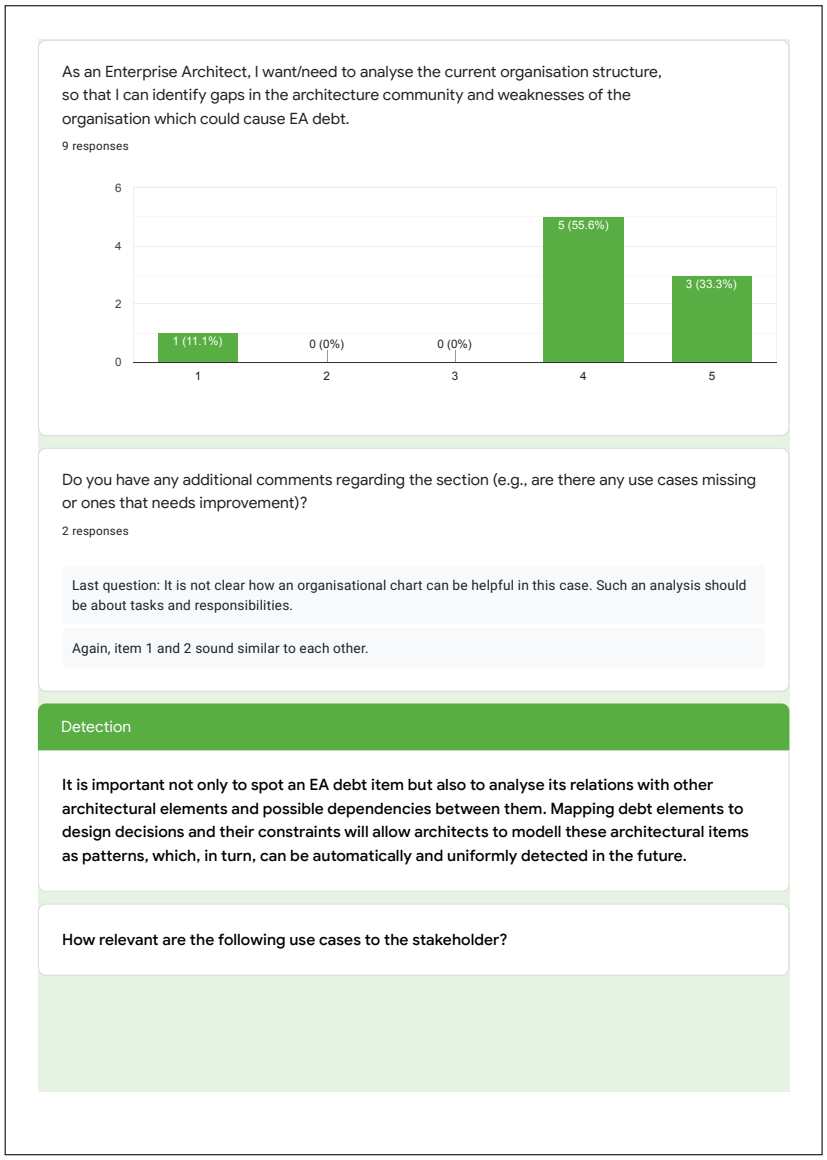

Figure B.10.: Questionnaire results page 10 of 15

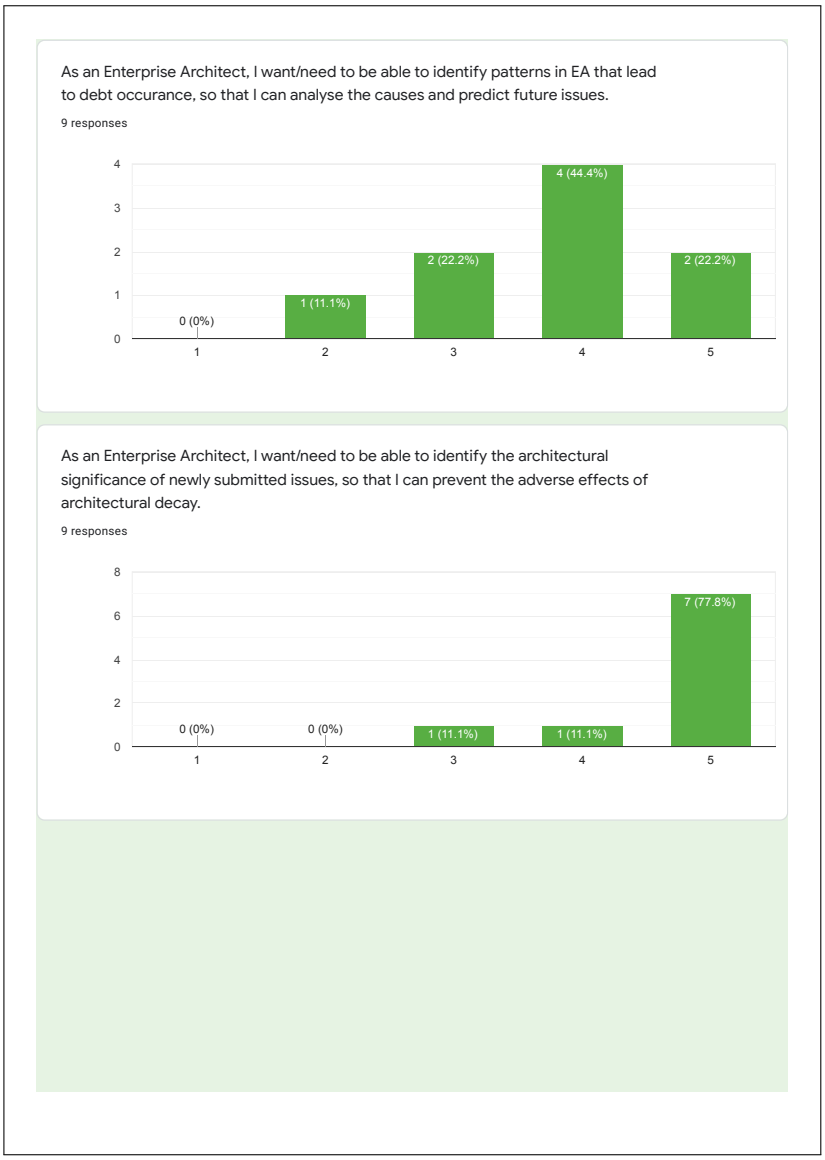

Figure B.11.: Questionnaire results page 11 of 15

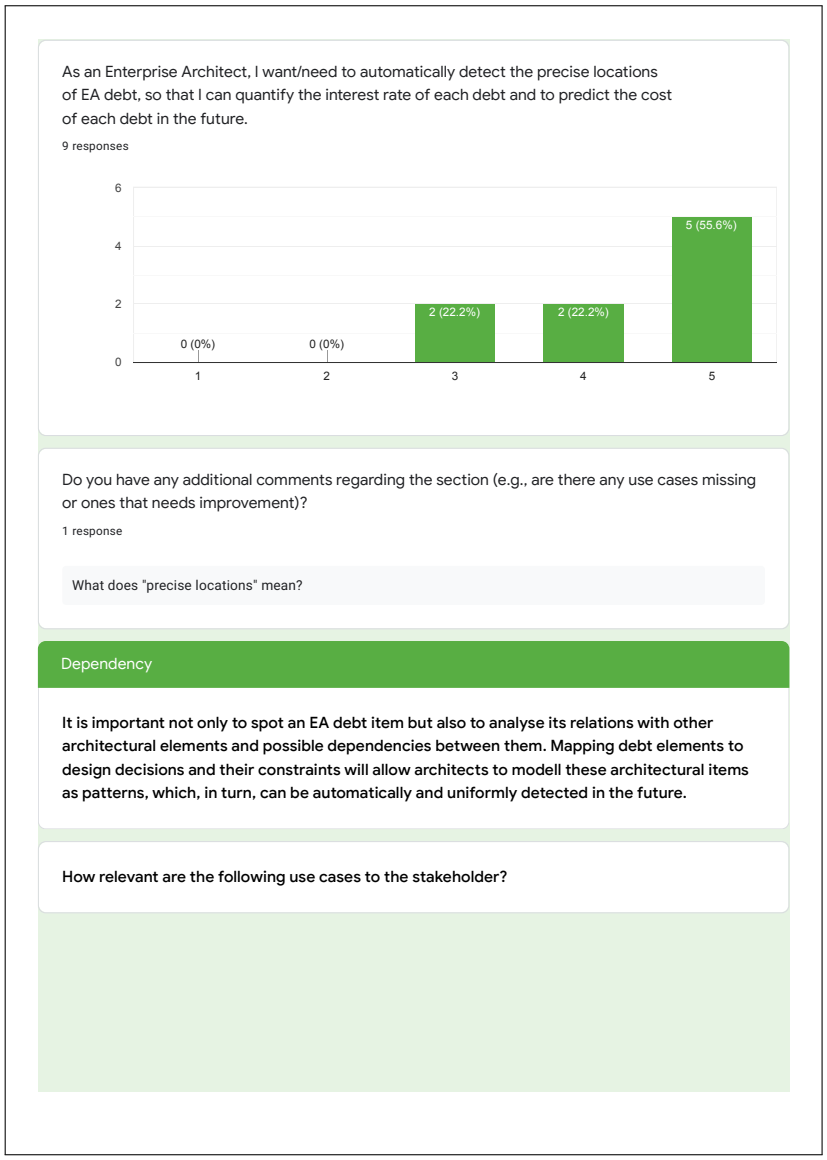

Figure B.12.: Questionnaire results page 12 of 15

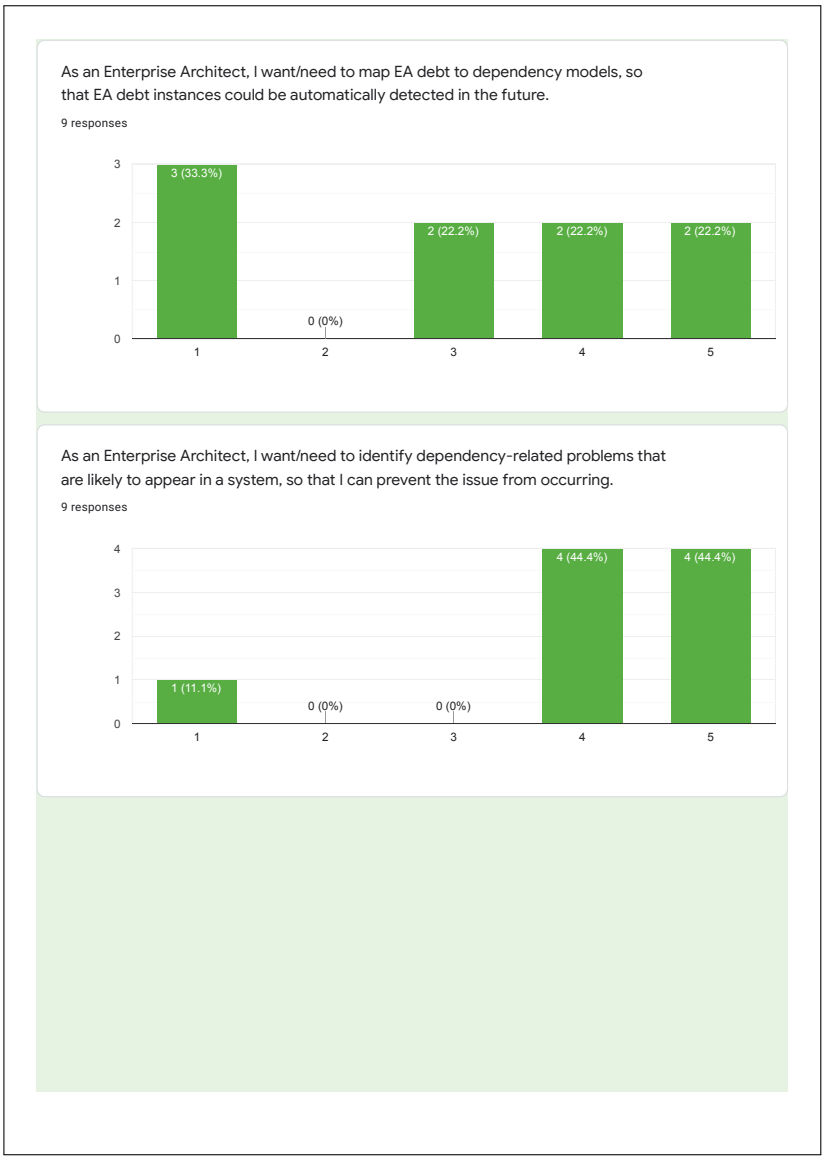

Figure B.13.: Questionnaire results page 13 of 15

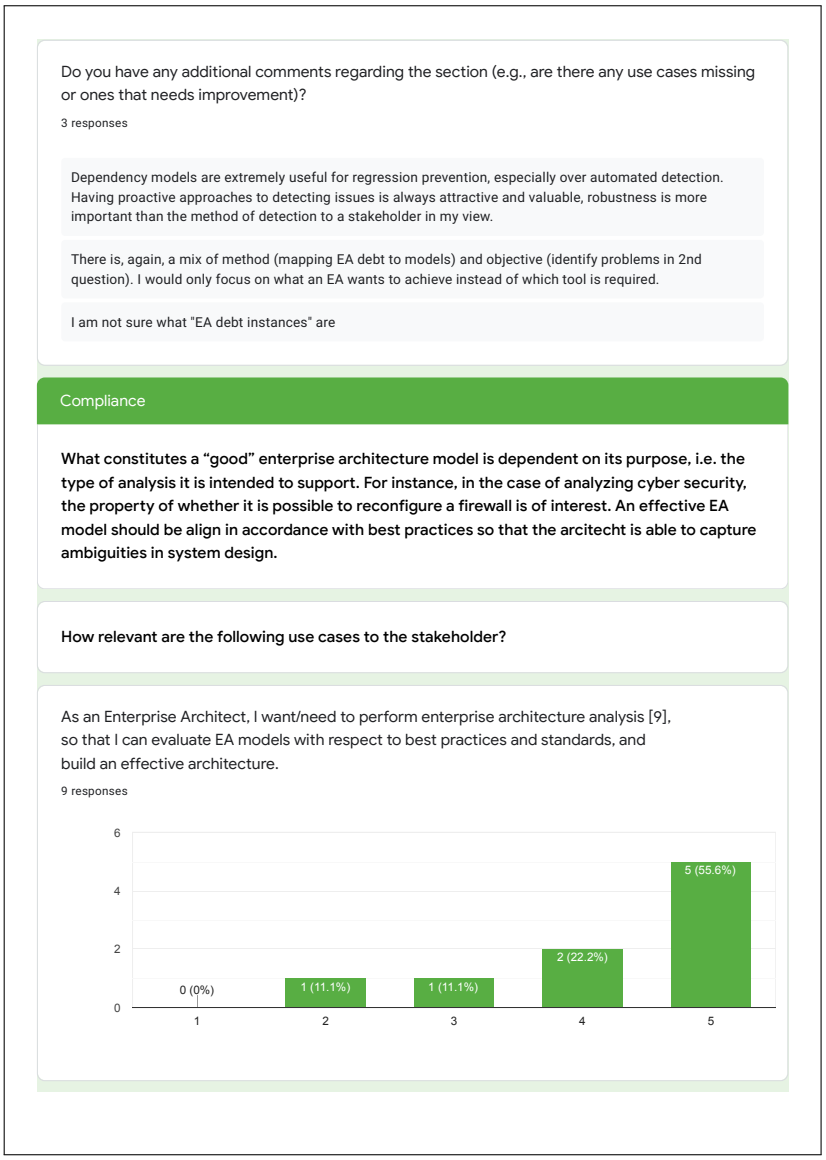

Figure B.14.: Questionnaire results page 14 of 15

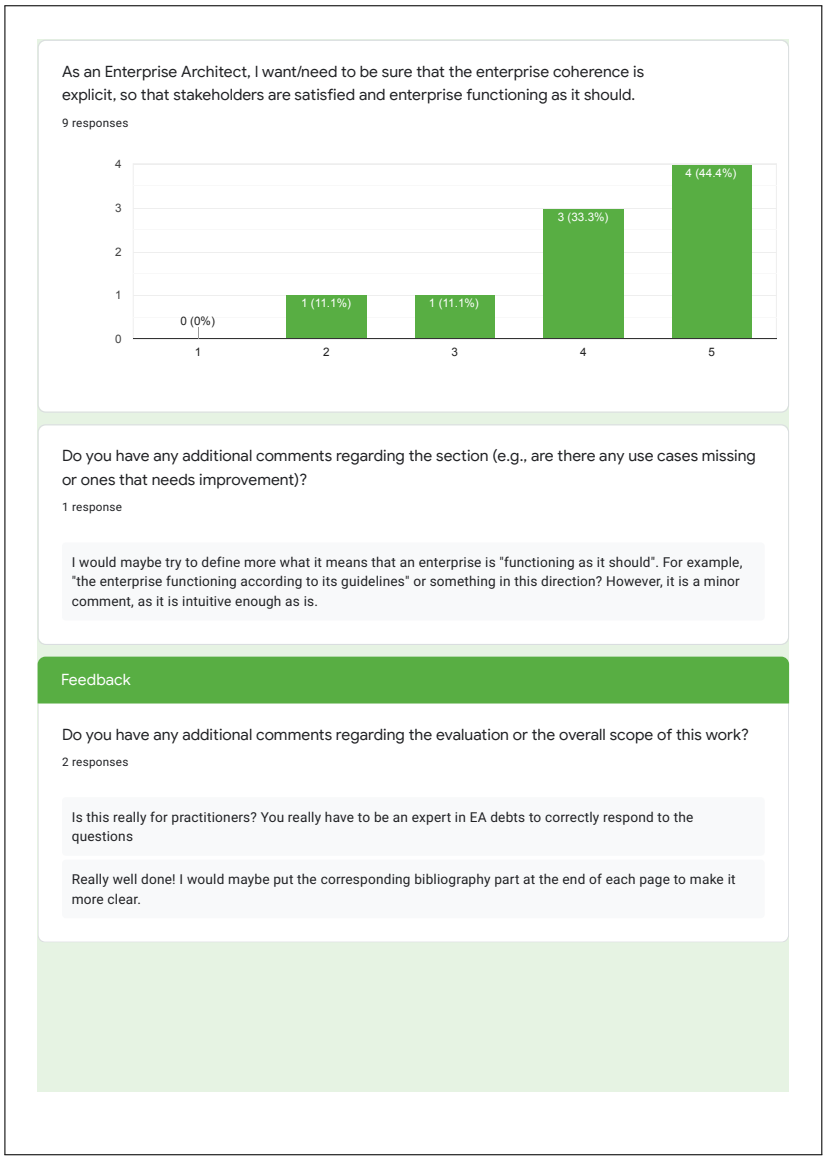

Figure B.15.: Questionnaire results page 15 of 15

Bibliography

[Ale+20] P. Alexander et al. “A Framework for Managing Enterprise Architecture Debts-Outline and Research Directions.” In: *EMISA*. 2020, pp. 5–10 (cited on page 12).

[BMB18] T Besker, A Martini, and J Bosch. “Managing architectural technical debt: A unified model and systematic literature review”. In: *Journal of Systems and Software* 135 (2018), pp. 1–16. ISSN: 0164-1212. DOI: 10.1016/j.jss.2017.09.025 (cited on pages 29, 39).

[BS19] N. Banaeianjahromi and K. Smolander. “Lack of communication and collaboration in enterprise architecture development”. In: *Information Systems Frontiers* 21.4 (2019), pp. 877–908 (cited on page 51).

[Cit] *Citavi*. https://www.citavi.com/en (cited on page 14).

[Cun92] W. Cunningham. “The WyCash portfolio management system”. In: *ACM SIGPLAN OOPS Messenger* 4.2 (1992), pp. 29–30 (cited on pages 10, 15).

[Die01] J. L. G. Dietz. “DEMO: Towards a discipline of organisation engineering”. In: *Eur. J. Oper. Res.* 128 (2001), pp. 351–363 (cited on page 1).

[FG10] G. Fairbanks and D. Garlan. *Just Enough Software Architecture: A Risk-Driven Approach*. Jan. 2010 (cited on page 10).

[Fow09] M. Fowler. *Technical Debt Quadrant*. https://martinfowler.com/bliki/TechnicalDebtQuadrant.html. 2009 (cited on page 11).

[Gro08] O. M. Group. *Business Motivation Model. Version 1*. http://www.omg.org/spec/BMM/1.0/. 2008 (cited on page 9).

[Gro09] O. M. Group. *OMG Unified Modeling Language, Superstructure. Version 2.2*. http://www.omg.org/spec/UML/2.2/Superstructure. 2009 (cited on page 10).

[Gro11] O. M. Group. *Business Process Model and Notation (BPMN), Version 2.0*. http://www.omg.org/spec/BPMN/2.0. 2011 (cited on page 9).

[Gro13] T. O. Group. *ArchiMate® 2.1 Specification*. The Open Group, 2013 (cited on pages 13, 17, 18).

[GS11] Y. Guo and C. Seaman. “A portfolio approach to technical debt management”. In: *Proceedings of the 2nd Workshop on Managing Technical Debt*. 2011, pp. 31–34 (cited on page 10).

[Hac+19] S. Hacks et al. "Towards the definition of enterprise architecture debts". In: *2019 IEEE 23rd International Enterprise Distributed Object Computing Workshop (EDOCW)*. IEEE. 2019, pp. 9–16 (cited on pages 1, 5, 11, 36).

[Hof+07] C. Hofmeister et al. "A general model of software architecture design derived from five industrial approaches". In: *Journal of Systems and Software* 80.1 (2007), pp. 106–126 (cited on page 21).

[Iaf] *Integrated Architecture Framework*. https://www.capgemini.com/resources/architecture-for-the-information-age/ (cited on page 1).

[JB05] A. Jansen and J. Bosch. "Software architecture as a set of architectural design decisions". In: *5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05)*. IEEE. 2005, pp. 109–120 (cited on page 21).

[JE07] P. Johnson and M. Ekstedt. *Enterprise architecture: models and analyses for information systems decision making*. Studentlitteratur, 2007 (cited on page 1).

[Kap+08] L. Kappelman et al. "Enterprise architecture: Charting the territory for academic research". In: (2008) (cited on page 5).

[KC07] B. Kitchenham and S. Charters. *Guidelines for performing Systematic Literature Reviews in Software Engineering*. 2007 (cited on page 13).

[LAL15] Z. Li, P. Avgeriou, and P. Liang. "A systematic mapping study on technical debt and its management". In: *Journal of Systems and Software* 101 (2015), pp. 193–220 (cited on pages 10, 15).

[Lan+10] M Lankhorst et al. "Trends in Enterprise Architecture Research". In: *5th Workshop Proceedings, Delft, The Netherlands*. Springer. 2010 (cited on page 1).

[LPJ10] M. M. Lankhorst, H. A. Proper, and H. Jonkers. "The anatomy of the archimate language". In: *International Journal of Information System Modeling and Design (IJISMD)* 1.1 (2010), pp. 1–32 (cited on page 7).

[Men] *Mendeley Reference Manager*. https://www.mendeley.com/ (cited on page 17).

[Met] *Troux Technologies: Metis*. http://www.troux.com/products/. 2010 (cited on page 18).

[Pro+18] H. A. Proper et al. "Enterprise Architecture Modelling: Purpose, Requirements and Language". In: *2018 IEEE 22nd International Enterprise Distributed Object Computing Workshop (EDOCW)*. 2018, pp. 162–169. DOI: 10.1109/EDOCW.2018.00031 (cited on page 7).

[Sch12] A.-W. Scheer. *Business process engineering: Reference models for industrial enterprises*. Springer Science & Business Media, 2012 (cited on page 18).

[SH93] S. H. Spewak and S. C. Hill. *Enterprise Architecture Planning: Developing a Blueprint for Data, Applications and Technology*. USA: QED Information Sciences, Inc., 1993. ISBN: 0894354361 (cited on page 6).

[SLML17] P. Saint-Louis, M. C. Morency, and J. Lapalme. "Defining enterprise architecture: A systematic literature review". In: *2017 IEEE 21st international enterprise distributed object computing workshop (EDOCW)*. IEEE. 2017, pp. 41–49 (cited on page 5).

[Sys] *IBM: System Architect*. http://www-01.ibm.com/software/awdtools/systemarchitect/productline/. 2010 (cited on page 18).

[SZ92] J. F. Sowa and J. A. Zachman. "Extending and formalizing the framework for information systems architecture". In: *IBM systems journal* 31.3 (1992), pp. 590–616 (cited on page 1).

[Tog] *The Open Group Architecture Framework*. https://www.opengroup.org/togaf (cited on page 1).

[Zho+20] Z. Zhou et al. "A systematic literature review on Enterprise Architecture Visualization Methodologies". In: *IEEE Access* 8 (2020), pp. 96404–96427 (cited on page 7).