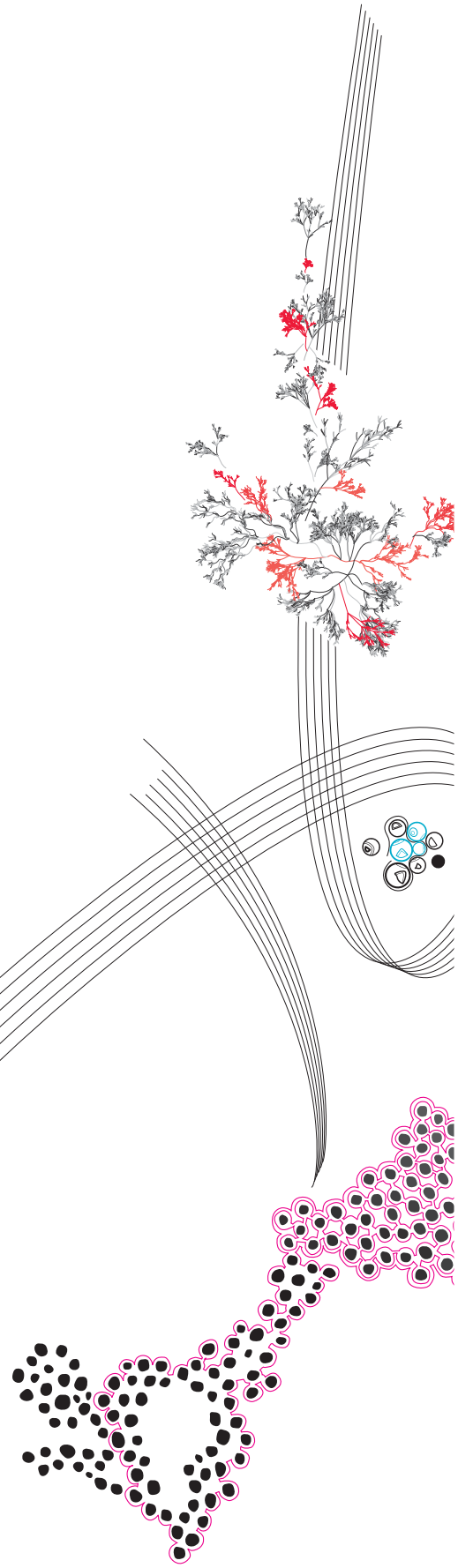


MSc Business Information Technology
Final Project

The Complexity of an IT Landscape

A Method for Quantifying Application Landscape Complexity and Understanding the Cost of Complexity

Eva Stoica

Supervisors:
dr. J.L. Rebelo Moreira
PDEng J.P.S. Piest

Enschede
July, 2024

Faculty of Electrical Engineering,
Mathematics and Computer Science

UNIVERSITY OF TWENTE.

Preface

For my bachelor's degree, I combined my passion for both IT and computer science with a human touch, by choosing Business and IT as my area of study. Throughout my bachelor's, I have discovered the field of Enterprise Architecture (EA) and I immediately fell in love with it. Now, for my master's degree, I wanted to delve deeper and explore how organisations use EA to promote a better understanding of IT for business purposes. Furthermore, something close to my heart is my experience as a teaching assistant. During the bachelor's and master's, I helped students in their learning journey, helping them understand complex topics and create effective study structures.

Thinking about these two aspects, I can easily understand some of the reasons behind my passion for EA. Enterprise Architecture is a means to make the invisible visible, allow people to comprehend different aspects of an organisation and create a coherence that enables achieving value faster in a dynamic environment.

For my master's thesis, I have decided to deepen the knowledge acquired and see how problems streaming from the dynamicity of today's IT landscape can be solved by applying an EA lens. And what would be a better way to do this instead of performing my thesis, in a real-life context, at a large company?

My master thesis journey spanned over 5 months in which I have learned so much about EA in practice. Furthermore, I delved into the challenges faced by large organisations due to the fast pace of technological advancements. Numerous people helped me during this time, without whom the thesis could not be this insightful.

I am extremely grateful for my university supervisors who guided, encouraged and supported me throughout my journey. Their feedback and incommensurable knowledge helped me deliver the best outcome possible. I would like to thank my company supervisors for being there for me every step of the way. They fostered the growth of not only my academic work but also my skills and mindset. Moreover, I am immensely appreciative of the enormous support received from the people involved in the case study. Based on their inspiring ideas and practical knowledge, spanning diverse research domains, the thesis evolved into its current form.

Last but not least, I could not thank my family and friends enough for their support and for their positive thoughts that always put a smile on my face.

I hope that all the heart that I have put into this thesis can have at least a small impact on the field of EA.

Eva Stoica

Hengelo, 5th of July, 2024

Executive Summary

Information technology (IT) is a field characterised by continuous and rapid transformations. In the context of large enterprises, numerous IT components are being used to allow the smooth operation of the business. Hence, IT represents a corporate asset that creates a strong relationship between business and IT. The number of components can become overwhelming and create a large IT landscape. This can be comprised of thousands of applications, configurations or proprietary developments, impacted by frequent and rapid changes. Hence, the overall complexity of such a landscape increases.

In the absence of adequate tools to handle this dynamic complexity and address its short or long-term impacts, companies struggle with the management and agility of the IT landscape. Given that complexity is an elusive quality characterising an IT landscape, it is essential to comprehend its origins, effective management strategies and the associated costs. Enterprise Architecture (EA), a discipline striving at the alignment of business and IT, plays a pivotal role in both managing and understanding the intricacies of IT landscape complexity.

To make the complexity of an application landscape (as a subset of the IT landscape) visible and manageable, as well as comprehend the types of costs associated with it, this research explores the design of a measurement method. The measurement should be applied when analysing EA models or designs as part of IT change projects.

A mixed-method literature review was performed to identify the drivers and metrics for IT landscape complexity, alongside the creation of an overview of IT landscape costs. Next, the indicators for application landscape complexity (ALC), how they influence each other and the growth of complexity were examined via a case study approach. These insights served as a basis for developing a measurement model and method, validated by employing experts' opinions.

The model for ALC identifies three key indicators, considered sufficient in assessing complexity. These are the number of interfaces, functionality overlaps and data overlaps among applications. Quantifying these indicators using EA models or designs provides a tangible complexity score. This score can help in making effective decisions for a less complex landscape. Additionally, this research emphasises various costs associated with complex application landscapes, including missed opportunities, rework, or prolonged decision processes.

As organisations struggle with managing the complexity of their application landscape due to rapid IT growth and increased business and IT interdependence, the proposed method comes as an aid. The artefact introduced helps in quantifying application landscape complexity for more effective decisions. This is an essential first step, before addressing business-driven complexity, towards achieving lasting agility and a simpler IT landscape.

Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
ALC	Application Landscape Complexity
API	Application Programming Interface
CMM	Capability Maturity Model
COC	Complexity Opportunity Cost
CP2	Cost Prediction of Change Propagation
DDC	Enterprise Architecture Degree of Dynamic Complexity
DSRM	Design Science Research Methodology
DT	Digital Twin
EA	Enterprise Architecture
IS	Information Systems
IT	Information Technology
ITIL	Information Technology Infrastructure Library
LoB	Lines of Business
M&A	Merger(s) and Acquisition(s)
MDM	Master Data Management
MECE	Mutually Exclusive Collectively Exhaustive
MSForms	Microsoft Office Forms
P2P	Point-to-Point
PRISMA	Preferred Reporting Items for Systematic Reviews and Meta-Analyses
ROI	Return on Investment
RPC	Remote Procedure Calls
RQ	Research Question
SCM	Structural Complexity Measure
SLR	Systematic Literature Review
SOA	Service Oriented Architecture
TC	Total Complexity
TOE	Technology Organisation Environment
TOGAF	The Open Group Architecture Framework

Chapter 1

Introduction

Chapter 1 introduces the research and the background information found at the core of the thesis development. This helps in creating an understanding of the investigated topic.

Section 1.1 establishes the context of the thesis. The next section, 1.2, describes the related works. In Section 1.3 the problem statement, objectives, scope and relevance of the research are detailed. This is followed by a detailed overview of the research questions and the guiding methodology, in Section 1.4. Lastly, Section 1.5, showcases the structure of each chapter of this thesis.

1.1 Context

The information technology (IT) landscape undergoes perpetual evolution, always being shaped by dynamic transformations [6]. Nowadays, companies are dealing with an overwhelming number of technologies that allow them to conduct their day-to-day operations [7]. Thus, IT represents a corporate asset, composed of numerous applications, configurations, interfaces, and proprietary developments, as part of a dynamic milieu characterised by frequent and rapid changes [8], [9].

These elements of IT and their dynamic relationships lead to an increasing complexity of the architectural landscape. Having insufficiently controlled growth caused by fast application development makes it difficult to manage the IT landscape efficiently [10]. Most current landscapes of enterprises are the results of a history of application implementation projects that introduced their specialised hardware, software, or infrastructure [11]. Over the years, enterprises have combined older technologies, or legacy applications, with newer ones [12], [13]. With advancements in technology such as digital twins, cloud computing, and big data analytics, the complexity of IT landscapes is increasing [14].

This happens as the IT architecture must address current business requirements, but also anticipate future needs. Mutual dynamism exists between business and IT. Changes streaming from one domain invariably influence the other. Sometimes, contemporary demands from the business side result in the addition of new solutions to the landscape that neglect to restructure existing systems despite similar purposes [15]. Also, the fast pace of developments can influence the prioritisation of short-term or quick fixes that lead to architectural technical debt [16], [17]. Technical debt refers to the design choices suitable when made, for a short-term payoff, however impending the progress towards a desired future state. Several factors such as the emergence of state-of-the-art applications to match

business needs or as implementing temporal measures (e.g., for data migration [18]) can therefore influence the dynamics of the complexity of an IT landscape and its development over time. Hence, a need for flexibility and agility are required such that the business functionalities will be supported by a variety of digital technologies [19]. These benefits are associated with "a right level of complexity" and an organisation can leverage Enterprise Architecture (EA) and methods, tools or techniques to manage complexity [4], [20].

The understanding of dynamic complexity, which refers to the interaction between components within the system and the change of the relationships over time, should help organisations manage the complexity of their IT landscape. Thus, dynamic complexity refers to the state changes over time and landscape transition, as opposed to the behaviour of systems during their execution. However, the literature giving focal attention to architectural complexity does not analyse dynamic complexity within large systems [2], nor look at the short and long-term impacts of complexity in an enterprise.

This absence of attention to dynamic complexity is also visible at a measurement method level, as little to no metrics exist to quantify dynamic complexity. Limited companies have suitable methods or tools to systematically assess and evaluate complexity [3]. This lack of tools is considered a major limitation to understanding the complexity levels of an organisation [21].

Furthermore, there is no means to value the "costs of complexity" for an IT landscape. The academic research is sparse and fragmented when describing how complexity in an IT landscape drives costs and what these exactly are. When a synthesis of the terms cost and complexity is created, most studies explain what type of costs can be calculated for the total IT landscape. This means applying a total cost of ownership perspective, thus only asserting the costs of acquiring, deploying, operating or maintaining the IT infrastructure and services over their lifetime. With this view, the costs similar to the one of subsequent changes are not addressed.

1.2 Related Works

To understand the situation described above, it is important to first acknowledge what the IT landscape is and how that intertwines with an architectural perspective. This is followed by grasping complexity through the prism of IT and showcasing how EA models can be leveraged in the management of complexity by using analysis techniques.

1.2.1 IT Landscape Architecture

Architecture

Starting with the term architecture, due to its eclectic nature, it is being used in a myriad of contexts and disciplines. It represents the philosophy which underlies a system and describes its purpose, intent, and structure [11].

Architecture defines the fundamental organisation of a system embodied in its components, their relationships to each other and the environment, as well as the principles guiding its design and evolution [4], [22]. Another well-acknowledged view on architecture is that it represents *"a pragmatic, coherent structuring of collection of components that [...] support the vision of the full user in an elegant way"* [23, p. 3].

The above-mentioned accepted definitions showcase that architecture is concerned with the

study of systems and their constituent elements alongside the ever-changing relationships involved in a system's composition.

Enterprise Architecture

The thesis investigates the IT landscape of large and complex organisations or enterprises. An enterprise can be defined as "any collection of organisations that has a common set of goals and/or a single bottom line" [24]. Architecture is an instrument which helps in controlling the complexity of an enterprise and its processes and system [4]. Hence, architecture, at the level of an organisation is referred to as enterprise architecture [4].

Enterprise Architecture (EA) represents *"a coherent whole of principles, methods, and models that are used in the design and realisation of an enterprise's organisational structure, business processes, information systems, and infrastructure"* [4, p. 3]. Moreover, it is a means to support the business and IT alignment [25] and to help with bridging the gaps between the current and the future states of an enterprise [24]. A better alignment between business and IT is believed to lead to "lower cost, higher quality, better time to market and greater customer satisfaction" [24].

There are three elements needed for Enterprise Architecture: methodologies, viewpoints, and modelling languages. TOGAF [24], [26] is a framework or a methodology that provides the methods and tools for assisting in the acceptance, production, use, and maintenance of an EA. It is based on an iterative process model supported by best practices and a re-usable set of existing architecture assets.

To implement such an architectural framework and create visualisations, modelling languages are needed. The most widely used language is ArchiMate [1] which complements TOGAF. From a modelling perspective, ArchiMate provides a metamodel combining layers and aspects to depict Enterprise Architecture. The core framework, as seen in Figure 1.1, can be considered an abstraction for the IT landscape, especially focusing on the application layer and its relationships upwards with the business and downwards with the technology layer. By combining business, applications and technology layers, different components can interact with each other via relationships. The components can also be passive, such as data objects, showcase behaviour, for example, application services or processes, as well as active structures which can be represented by interfaces. Their combination creates a complete, integrated approach to delivering enterprise architecture.

IT Landscape

Enterprise architecture planning is thus the ground for the development of the IT landscape of an organisation. It provides numerous benefits such as the improved agility to react fast to the market requirements [25], or to manage the complexity of an enterprise.

An IT landscape can represent *"A set of hardware, software and facility elements, arranged in a specific configuration, which serves as a fabric to support the business operation of an enterprise"* [27, p. 383]. However, no standard definition of such a landscape exists [11], [23], [28]. In literature, IT landscape can be used interchangeably with concepts such as application architecture, IT infrastructure, technical architecture or information systems (IS) architecture.

Hammer describes IT architecture as a common high-level model or design that supports various disciplines and stakeholders in *"taking design decisions in a multi-dimensional design space"*, *"defining designs in a multi-disciplinary environment"* and *"communicating so-*

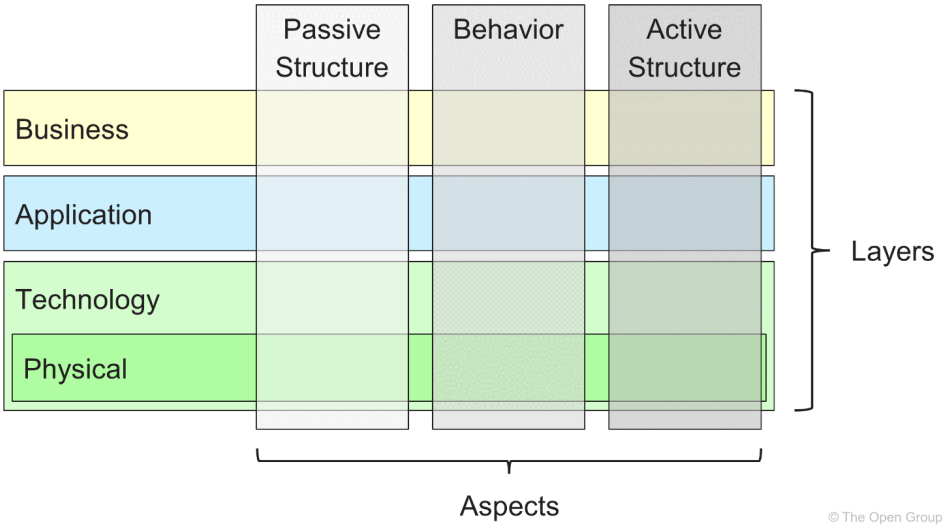


FIGURE 1.1: ArchiMate Core Framework [1]

lutions" [28, p. 305]. IS architecture can be interpreted as such: *"the overarching structure and properties of the relationships among the systems and applications in an organisation's IT portfolio"* [29, p. 1].

Therefore, the consensus behind the meaning of IT landscape (used as the basis for the thesis) is the use of various interrelated artefacts, such as application systems, that enable the provision of services to the business function or support business processes or capabilities [11],[25],[30]. This ultimately leads to more informed decision-making processes and better communication.

To observe these interrelationships, a conceptual model of an IT landscape and its components was introduced in the previous work of the researcher [31], after piecing together findings from prevalent academic works. Figure 1.2 depicts how from the business requirements, there are various applications with certain characteristics that support these demands. Appendix A, contains the figure's detailed description.

Architectural Decisions

When designing an IT architecture, an organisation can make various decisions for representing their landscapes. These are concerned with elements such as architectural styles, approaches for data management, infrastructure and deployment, or security and compliance [32], [33], [34], [35], [36].

Two types of architectural decisions are relevant to this thesis, the choice of architectural styles and the approaches for data management.

Starting with architectural styles (or patterns), these describe ways to create and structure objects and their behaviours [37]. For example, microservices are architectural styles that focus on the development of a single application as a suite of small services, each running its processes and communicating with lightweight mechanisms [32]. This is opposed to having a monolithic application whose functionality is encompassed into a single process. Other elements in "a catalogue" of architectural patterns can be the use of service-oriented architecture [38], client-server architecture [39] or event-driven architecture [40].

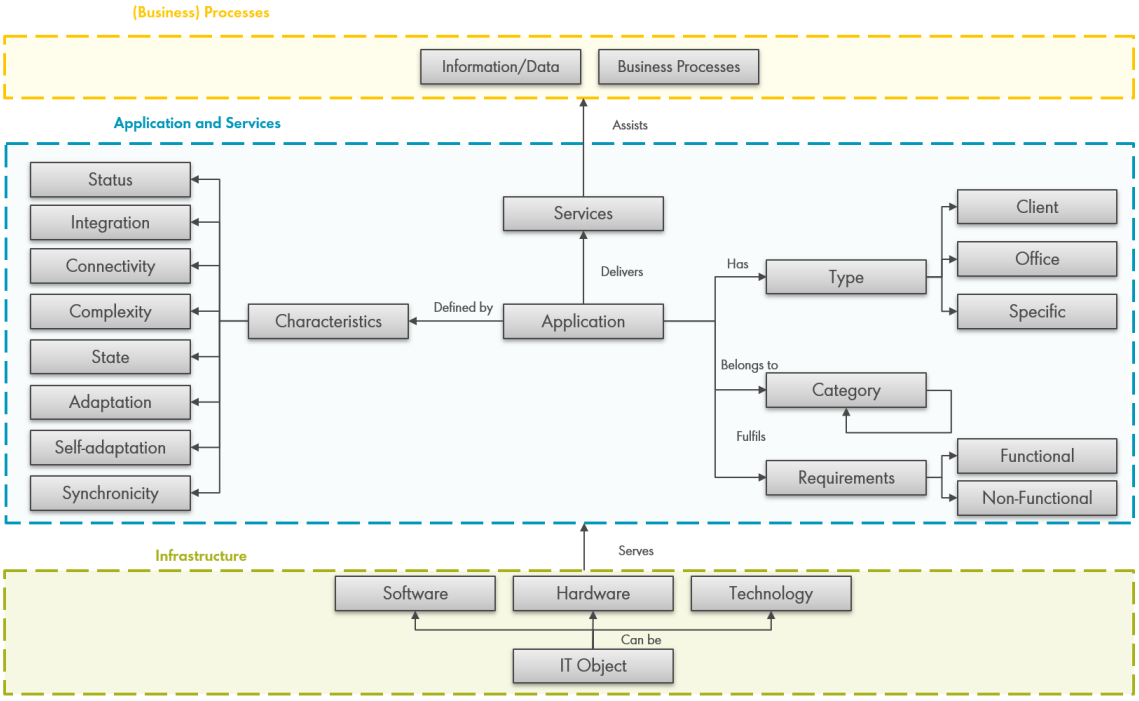


FIGURE 1.2: IT Landscape Components

Next, other decisions such as how to distribute, store and integrate data between IT components can influence how a landscape is represented. Nowadays, discussed concepts for data management are the use of data warehouses or lakes (as formats for storing, analysing and querying data) and data meshes (as a data management approach). These can be classified under the umbrella of "data integration architectures", as explained by Wrembel [33].

Data warehousing represents a process connecting and managing data from different sources to gain a single, detailed view of parts or all of a business [41]. Data integration is performed after the data is transformed and according to a common data model, cleaned, normalized and stored in a repository [33].

Data lakes are platforms for big data management which store and analyse raw data [42]. A data lake architecture and its composing elements, as well as a methodology for its configuration, are proposed in the literature by Giebler et al. [43]. The difference between a data warehouse and a lake is that the first is a repository which stores structured and processed data assisting a specific purpose, whereas the second is managing raw data with an undefined purpose [42].

Lastly, data meshes describe the approach of decentralising data (as opposed to the two approaches mentioned above which are centralised) and creating repositories that can be organised or used by specific business domains [33], [44]. This means each domain has autonomy over its data, which is seen as a product.

Therefore, when designing IT architecture, a myriad of decisions can shape the overall structuring and behaviour of the landscape. These range from high-level design strategies such as architectural styles, towards more technical ones as seen with the selection of data management approaches.

1.2.2 Complexity

IT Landscape Complexity

In the last years, complexity has become a buzzword which is applied in various domains such as architecture, biology, chemistry, computer science or political and social sciences [45], [46]. IT landscape complexity refers to the number of components or elements of an architecture, their relationships and the variation or heterogeneity of these [47]. Onderdelinden et al. [18] mention that the IS architecture complexity must also include different levels of observation. This is important as architecture can be seen as a whole or at a granular level, or influenced by the nature and rate of change of the components and relationships present.

For the field of information systems or IT landscape complexity, Schneider [2] unified the existing research by creating a multi-dimensional framework, as depicted in Figure 1.3. Complexity can be defined according to four dimensions. These dimensions are aligned with complexity research streams and encompass the most important and well-acknowledged findings. Furthermore, newer academic findings (e.g., [48]) can be structured according to Schneider's work. Hence, this framework represents one of the best classifications for IT complexity.

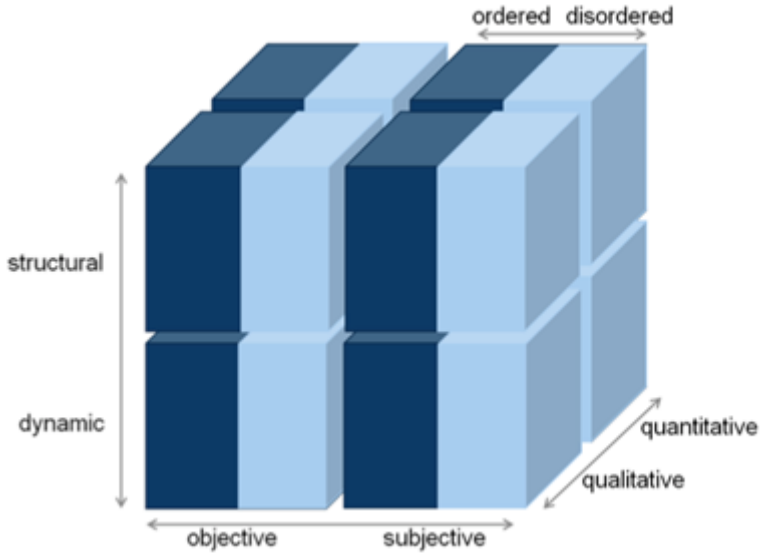


FIGURE 1.3: Complexity Cube [2]

1. **Organised vs. Disorganised**
   The first dimension reflects organised and disorganised complexity, as also proposed by Weaver [49]. In 1948, the author first described these two views. On one hand, disorganised complexity means having many variables, each with an individual erratic or unknown behaviour, as part of a system with *"orderly and analyzable average properties"* [49, p. 538]. On the other hand, organised complexity deals with *"a sizeable number of factors which are interrelated into an organic whole"* [49, p. 539].

2. **Qualitative vs. Quantitative**
   Moving on, the second dimension is qualitative and quantitative complexity. The first pillar of this dimension refers to how evaluations of elements are performed in a qualitative manner, whereas quantitative complexity is the assessment based on

numbers (e.g., Kolmogorov complexity [50]).

3. **Subjective vs. Objective**
   The third dimension aims at differentiating subjective and objective complexity. This means that the role of the observer is essential. Objective complexity is independent of the observer. Subjective complexity discusses how a system is perceived by an individual. This is also exemplified in Fioretti's work [51].

4. **Structural vs. Dynamic**
   Lastly, structural and dynamic complexity are proposed. Structural complexity looks at the components of a system and the cause-and-effect relationships between them. Dynamic complexity showcases how components and their interrelationships are changing over time, from one state to another.

   The dynamic complexity can be further broken down into short-term and long-term complexity [5]. Short-term refers to sudden and unpredictable changes, whereas long-term includes extended time scales for which plans are in place.

Complexity Management

These numerous prisms, through which complexity can be looked at, can be applied in the context of enterprises. Nonetheless, to grasp complexity, and turn the invisible into visible, companies are searching for ways to understand its size and impact [52].

Complexity management is concerned with a myriad of tasks that can potentially make the transition from intangible, hard-to-recognise issues towards visible ones. The tasks under the umbrella of "complexity management" range from considering and solving problems resulting from the variety or dynamics of elements in an IT landscape to integrating measures to tackle problems into frameworks [53].

Kluth et al. [3] explain that delaying with complexity is a competitive factor nowadays. Complexity is seen as the *"combination of various, heterogeneous system elements and their interrelationships that are changing dynamically and not clearly compressible"* [3, p. 226]. The trend of increasing digitalisation allowed only a few companies to have abilities that can help with tracking and managing complexity. *"Only 16% of the consulted companies state, that they have suitable methods or tools to systematically assess and evaluate complexity"* [3, p. 225].

To address this issue, a maturity model for the assessment of complexity capabilities of an organisation is proposed, as seen in Figure 1.4. The model is inspired by the well-known Capability Maturity Model (CMM) [54], which *"has already been used in more than 3,000 companies"* [55, p. 506]. Kluth et al.[3] describe seven steps to assess the complexity management skills of an organisation, as seen in the figure below. Based on the complexity capability maturity model, an enterprise can observe its current positioning and discover how to improve its present standing.

1.2.3 Enterprise Architecture Analysis

EA Analysis

When managing complexity, and when changes must be made in the IT landscape of an enterprise, model-based analyses play a pivotal role for an organisation. This is especially the case for large enterprises in which the size and complexity of architectures can no

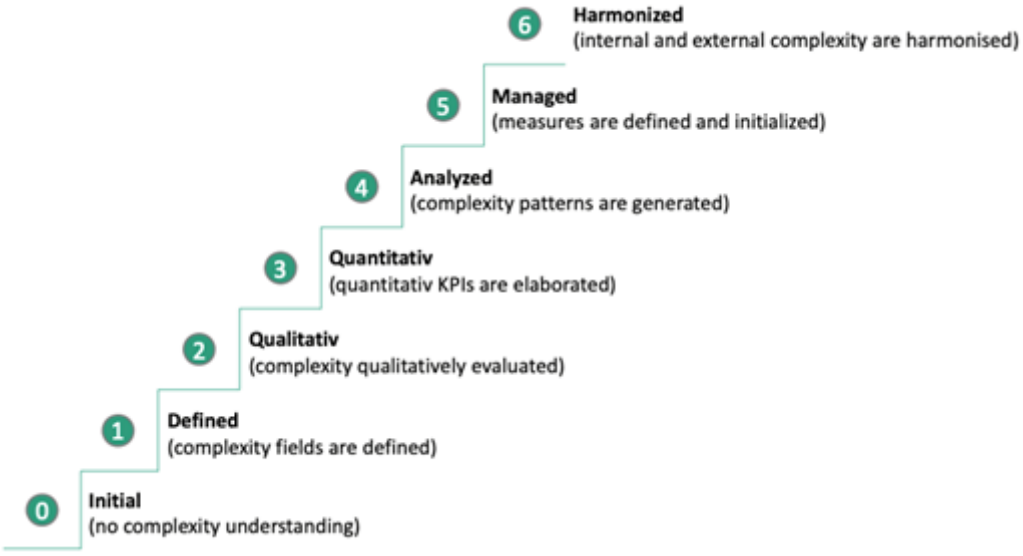


FIGURE 1.4: Complexity Maturity Model [3]

longer be identified by hand. To create a comprehensive view of EA, both descriptive and model-based analysis techniques should be leveraged [4].

There are several architecture analysis techniques and approaches. These can be classified according to two dimensions: the type of analysis inputs and results or the type of technique [4], as seen in Figure 1.5.

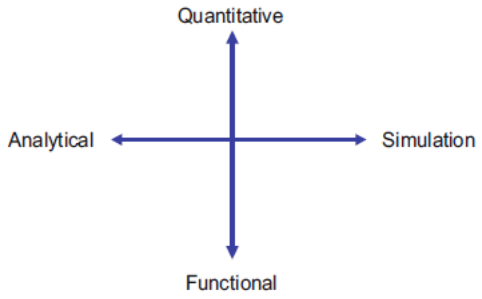


FIGURE 1.5: Analysis Dimensions [4]

The first distinction is based on the analysis input or results. These can be either functional or quantitative. The functional analysis addresses how the system works and what are its properties (e.g., structural properties). The quantitative analysis aims at uncovering aspects such as the performance, cost or quality of a system.

One of the most well-known approaches for performing quantitative analysis of EA models, realised with ArchiMate, is introduced by Jonkers et al. [56],[57]. This is essential in understanding how performance or workloads can be quantified.

Next, for the type of techniques division, there are two pillars: simulation and analytical techniques. This means that for functional or quantitative analysis one can perform analytical techniques or simulations to gain insights based on EA models. Simulation refers to the execution of a model in the sense that *"architects can play"* [4, p. 200] with the

architecture and create statistical statements about quantitative measures after multiple model runs. The analytical techniques are concerned with the production of a reproducible result, therefore not based on statistics.

The analytical techniques combined with the quantitative input are considered essential for comparing various alternatives or what-if situations [4].

Measurement Methodology

To strive towards the quantification of complexity and perform a model-based analysis it is also important to understand what such a process entails in terms of measurement. According to metrological [1] definitions, complexity can be classified as a latent variable. This means that additional, observable or measurand variables are needed to quantify this concept. From an enterprise architecture perspective, complexity is an attribute (or a quality of the model) that should be measured with the help of metrics.

To measure the variables, one or more metrics need to be in place. The metrics need to be expressed as numbers and, thus be measurable. Numerous metrics can exist, thus, when constructing a measurement, the requirements of stakeholders must be considered. This is important when selecting an appropriate amount that aids in indicating the value of the attribute. In the literature, no specific guideline exists on how many metrics or factors should be included in a measurement. For factor analysis techniques, such as exploratory factor analysis (which focuses on the appropriate selection of observed variables), a suggestion is made to select four to eight indicators per factor [58]. Furthermore, Aigner [59] mentions the selection of the top three elements from an ordered list to identify a concept. For formula creation guidelines, Krynicki [60] also supports the use of a few quantifiers. In scale creations, better precision is observed by using three to four items [61]. Furthermore, for architectural decisions, selecting from a list of three to seven concerns must be the focus of the architects, for significant results [62].

Measuring architectural complexity is the process that uncovers the value of the metrics that comprise complexity. The origins of complexity must be identified, and measurements for each should be detailed. Furthermore, a method for operationalisation of complexity should follow.

According to Abran [63] the above-mentioned steps, fall into two categories: the measurement principle and the measurement method. The measurement principle defines the object of interest and the metrics connected to it, in the case of the thesis, complexity and potential metrics. Next, to measure the attribute, e.g., complexity, steps that need to be followed are described in the measurement method.

For the context of the thesis, both a measurement principle and a method for quantifying complexity are proposed. It is important to note that the measurement should be limited to a few metrics selected with the help of company stakeholders, to make it implementable/usable in the practical setting of an enterprise.

Table 1.1 details the terminology used throughout this research when it comes to measurement of complexity. It can be seen that an attribute is composed of multiple indicators, which can in turn be measured by using numerous metrics.

[1] Metrology - the scientific study of measurement

Table 1.1: Measurement Terminology

Concept	General Terminology
Complexity	Attribute/Latent variable
Driver of complexity	Indicator/Observable variable/Measurand/Factor
Quantifying the driver	Metric

1.3 Research Objectives

This section starts by describing the problem statement and the thesis objectives. Next, the scope and relevance of the thesis are detailed.

1.3.1 Problem Statement

Given the IT growth and the dynamism of the field, managing the complexity of an IT landscape in an organisation can be considered troublesome. The problem is further amplified by the close connection between IT and business. This means that IT changes when the business transforms and vice versa.

This complexity of IT landscapes is an intangible factor and most organisations experience difficulties in understanding its size and its impact [64], [65]. Measuring the complexity linked to a change, or rather if a presently higher complexity of an IT landscape imposes more difficulties in achieving a desired outcome, can, in consequence, be perceived as a very challenging endeavour. To address these issues, using an architectural representation could serve as the lens for making the invisible visible.

Architectures should keep in mind the "requisite complexity", meaning finding a balance between excess and deficit complexity, such that a high value can be gained [66]. In other words, every system must have a minimum level of complexity *"to deal with requirements stemming from the system environment"* [18, p. 2].

Due to the close connection between business and IT and the frequent changes (if one changes the other does as well), finding the right level of complexity that permits flexibility is needed. This is important as business requirements need to be addressed promptly by IT.

Furthermore, this right level of complexity in an IT landscape is also crucial and has a significant impact on the economic pillar of an organisation, amongst other elements. A sharp increase in IT complexity is often accompanied by higher costs [15]. Large enterprises try to tackle the issue of complexity, by using various programs aimed at simplifying the IT landscape. These can include initiatives such as creating architectural archetypes or remodelling current architectural representations. They aim to achieve a cost reduction or a positive effect on the economic pillar of a company.

Beyond costs, managing complexity has implications for factors such as agility, flexibility, risks, or efficiency [20]. The adaptation in a timely manner to the dynamic environment should be investigated. This research is focused on the impact of complexity on agility, as organisations need to adapt quickly in the dynamic milieu characterised by frequent IT changes. Complexity can be seen as the inability to change or efficiently adapt to a rapidly evolving environment.

Hence, the problem statement can be encompassed as follows. Due to the growth and frequent changes in IT, managing complexity and comprehending its associated costs is posing a challenge for large organisations striving to be agile.

1.3.2 Objectives

Based on the problem statement detailed previously, it is evident that quantifying the complexity of the IT landscape is an important step that will support the creation of processes for the management of complexity. With the help of the management of complexity, adapting to the dynamic environment in which a company operates could become an easier endeavour.

Understanding how to create a measurement method that can assist in complexity management approaches could reap benefits for large enterprises in today's rapidly changing environment.

Main Objective

The key objective of the thesis is to define a measurement method for calculating a subset of the IT landscape complexity, the one associated with application landscapes. The method should be applied in a practical setting to support companies in mastering complexity. As IT can look at the use of various interrelated artefacts such as application components, physical infrastructure, and technology, the scope was limited to understanding the applications part of the IT landscape, their relationships and data exchanges.

This should be done by comparing and contrasting various architectural representations. These must be represented graphically or documented in a specific manner, such that initial and target states' designs can be the basis for quantifying the application landscape complexity (ALC).

Sub-Objectives

To achieve the main objective, several sub-objectives have been created.

1. Review state-of-the-art literature to understand which complexity metrics exist.
2. Review state-of-the-art literature to understand the relationship between costs and complexity.
3. Select and categorise relevant complexity indicators and metrics with the help of a case study performed in a large enterprise.
4. Create a measurement method that can be applied to an organisation's application landscape to help guide decisions.
5. Discover what are the costs that can be encountered when ALC hinders the agility of an organisation to adapt to changes.

1.3.3 Scope

Given the dynamics of today's world, especially when discussing about IT developments and growth, enterprises need to have the ability to quickly adapt to changes. Leveraging EA can assist an organisation in managing and understanding its IT landscape complexity. As

it was observed, complexity, especially the dynamics aspect, is insufficiently investigated, and large organisations do not have the necessary means or tools to quantify IT complexity.

To create a measurement method for quantifying the IT landscape complexity, the scope of this study is limited to the application and data elements of architectural representations, further called the application landscape. The method is initially designed as a means to evaluate ALC in the projects of an enterprise.

Projects are expected to depict the current state (as-is), as well as showcase the possible options for future change (to-be scenarios). Additionally, rather than attempting to quantify the entire IT landscape of an enterprise, projects facilitate a more efficient information gathering. They are also the moments in organisational changes are implemented, thus, they are the appropriate mechanism to influence strategic decisions. When the ALC will be quantified in projects, the involvement of a limited number of stakeholders allows for an easier and more precise calibration of the score.

Based on the as-is and the to-be representations, each design can receive a complexity score, and then the delta in the change from one state to the other can be calculated. This difference is essential, as it can serve as a choice for an option, based on the overarching assumption that less complexity implies an easier change process. For dynamicity, this can mean if the complexity levels increase when changing the landscape, less agility and more costs can emerge.

Moreover, as the complexity measurement is based on a selected array of metrics, finding what types of costs can emerge due to the presence of the selected indicators in the landscape is desired.

To explain what falls out of the scope of the thesis, the measurement method does not apply to causes and indicators of complexity that can be associated with the business environment or the infrastructure and the technology components part of an enterprise.

The study uses academic literature and information gathered from a case study performed within a large global organisation. These are detailed in the subsequent chapters.

1.3.4 Research Relevance

Based on the thesis, valuable and novel insights for academia and practice are expected to emerge. The overarching goal helps with a first step in creating a better understanding of a very intricate area of research: the IT landscape complexity. Proposing a complexity measurement method that can assist with enterprise architecture analysis introduces several innovative and previously unexplored perspectives.

From an academic standpoint, the study expands the body of knowledge on both complexity management and measurement. Well-acknowledged complexity metrics are fused into a formula, and a measurement method is proposed. Dynamic complexity is also addressed, as an EA model-based analysis is leveraged to investigate the current and future states of landscapes. Furthermore, a more in-depth link between complexity and costs is created, while identifying complexity metrics.

For practitioners, the paper raises awareness of the importance of adopting a maturity model for managing capabilities for complexity, alongside the need for a measurement for complex IT landscapes. As identified, currently, tools and techniques that can help with complexity understanding are mostly absent in organisations. Therefore, the measurement proposed which can be applied to EA models to analyse the size and cost of complexity,

is expected to help practitioners. During projects, the measurement can be used as a tool for option choices. Based on the level of complexity, more informed decisions can be made when making architectural changes. On the basis of such an analysis, practitioners can observe if complexity in their architecture is indeed an indicator of ease of change or adaptation to new developments or demands in the future.

1.4 Research Design

In this section of the report, the questions which were created to guide the research are introduced, alongside the manner in which an answer to these questions is composed.

1.4.1 Research Questions

One main research question (RQ) was formulated based on the previously identified problem statement, such that the thesis could have a clear direction. The question was broken down into several sub-questions to ensure that sufficient knowledge is accumulated and that new information can lead to interesting and unique insights.

Main Research Question

How can a practical measurement method be designed to quantify the application landscape complexity and its cost implications?

Sub-Research Question

1. Which complexity metrics can be used to measure change in an IT landscape?
   (a) How can complexity be classified?
   (b) What factors drive the complexity of an IT landscape?
   (c) What are the existing metrics for measuring architectural complexity?
2. What types of costs are associated with the IT landscape?
   (a) What Enterprise Architecture model-based cost analysis techniques can be applied to IT landscapes?
   (b) What are the prevalent usages of "cost of complexity"?
3. What are suitable complexity metrics that can be used in measuring the delta of the change in the application landscape?
4. What costs can be factored in a formula when application landscape complexity hinders the agility to adapt to a change?
5. To what extent can complexity metrics be used in analysing enterprise architecture designs in the context of a large enterprise?

1.4.2 Research Methodology

To create a measurement method that can quantify complexity by analysing of EA models or designs, the Design Science Research Methodology (DSRM) [67] can be employed. This type of methodology allows the design and investigation of artefacts in contexts. The

design problem is the issue that needs to be solved by means of an artefact. According to the DSRM, the design problem at the heart of the thesis is the following:

Improve	*the management of IT landscape complexity*
by (re)designing	*a measurement method*
that	*can be applied to quantify the application landscape complexity and its cost implications in a practical setting, based on documented designs of EA project solution options*
in order to	*help IT architects make effective decisions to keep the IT Landscape agile for future changes*

As a design cycle structures the thesis activities, the following steps must be performed: problem investigation, treatment design and validation. To support the design cycle, a case study [68] is conducted within a large organisation.

Under the problem investigation, the knowledge problems are addressed. In this initial phase, an exploration of the problem occurs prior to the design of an artefact according to certain requirements. Next, the first two sub-research questions help set the basis of the problem investigation. For constructing a foundation of existing knowledge and for crafting a holistic view that could help in answering the research questions, a combination of both exploratory and systematic literature reviews (SLR) is employed.

Nonetheless, it is essential to gather information from the enterprise involved in the case study, for which the measurement method is initially developed. A survey was designed to understand the current complexity capabilities of this enterprise, identify complexity drivers and rank the literature-identified metrics. Hence, during the problem investigation, two types of data collection techniques are used literature reviews and a survey.

The second step in the DSRM design cycle, the treatment design, focuses on the requirements for the measurement of complexity and how a method can help with the problem identified a priori. During this step, interviews with several stakeholders were conducted such that a prioritisation of the literature-identified complexity indicators could be in place. This links back to the third sub-research question. Two types of interviews were conducted, one emphasising the complexity of an IT landscape and the second discussing possible ways to measure the cost of complexity.

For operationalising the formula, the requirements needed for a measurement method must be understood. In this step, several iterations are done by the researcher to combine the identified indicators of complexity in a measurement model. At the same time, the researcher started designing a method that can be applied in the context of the enterprise in which the case study is conducted. Therefore, during the treatment design, a measurement method for applying the quantification of ALC for EA analysis is created.

Next, the measurement method should be validated. This is achieved by employing expert opinions. It is important to observe the possible effects of the combination of the artefact in context, as well as check if the requirements are met. Validation is also expected to illustrate possible trade-offs or sensitivity for different contexts.

Figure 1.6 depicts the steps described above, the task and sub-tasks assigned to each of them, and the addressed research question.

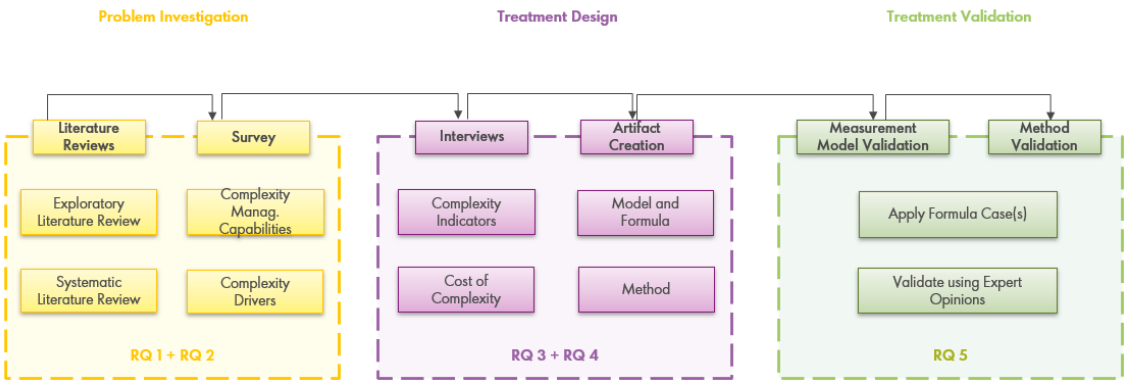


FIGURE 1.6: DSRM Process

1.5 Research Structure

To understand the situation, complication and the approach to answering the RQs, the document is structured in 7 chapters. Chapter 1 introduces the topic and the necessary background information that can make one familiar with the subject matter. This chapter also details the research methodology used. Chapter 2 describes the findings of a thorough literature review, addressing the topics of complexity metrics and costs associated with the IT landscape. Moving on to Chapter 3, this encompasses the description of the performed case study. The survey and interview structures and responses are described. Chapter 4 is illustrative for a detailed explanation of the formula and method designed to quantify complexity. Later in Chapter 5, the validation of the artefact is studied. Chapter 6 is dedicated to the discussion of the overall results, implications and findings. Lastly, Chapter 7 wraps up the performed study and offers the conclusions of the work.

Chapter 2

Literature Review

The second chapter delves into the findings collected through extensive literature analyses. The different approaches used to examine academic works are described in Section 2.1, and then, the procedures that guided the search are detailed in Section 2.2. The findings of the prevalent works related to complexity metrics and costs connected with the IT landscape are discussed in Section 2.3. Here, detailed answers are provided for the first and second sub-research questions. The chapter ends with a summary and conclusions, as seen in Section 2.4.

2.1 Review Methodology

To shed light on existing complexity drivers, possible ways to operationalise them, and understand what the cost of complexity can mean, a qualitative method of enquiry was used. A combination of both exploratory and systematic literature reviews (SLR) was employed. A literature review allows one to keep up with the state-of-the-art and to be *"at the forefront of research"* by collecting evidence for a particular field of interest [69, p. 333]. By having a mixed-method approach, a foundation of existing knowledge and a holistic view of the investigated topics was created.

2.1.1 Exploratory Literature Review

The goal of an exploratory, or unstructured review, is to investigate the vast array of articles that are linked with various research questions. The results create a range of possible answers (e.g., different options as to what something means). This approach does not intend to offer a conclusive solution or answer to a research question, as it rather explores the available information on the topic [70].

2.1.2 Systematic Literature Review

A SLR represents *"a means of evaluating and interpreting all available research relevant to a particular research question, topic area or phenomenon of interest"* [71, p. 3]. As opposed to an exploratory literature review (described above), a SLR emphasises rigorous, explicit, and reproducible methods, alongside a high level of transparency. To achieve this, guidelines for conducting a systematic literature review were consulted.

2.2 Review Process

To find an appropriate answer to the knowledge research questions (RQ 1 and RQ2) and their associated sub-questions, a deliberate decision on which type of literature review to use was made. In this part of the thesis, first, the decision is introduced, followed by the guidelines of the exploratory and the systematic literature reviews.

2.2.1 Knowledge Questions

The basis of the information-gathering process of this research lies in finding answers to the predefined research questions. The first two can be classified as knowledge questions. This means that they support the creation of a knowledge base upon which an artefact can be designed.

To provide an answer to the first knowledge question, "Which complexity metrics can be used to measure change in an IT landscape?", it was essential to explore existing classifications and meanings of complexity. As complexity is a recurring topic in numerous fields of study, having one classification of complexity that could further guide the research was important. A SLR was performed to gain a better understanding of the factors influencing the complexity of an IT landscape and narrow down the elements influencing the application layer. Once the drivers were identified, existing metrics that could operationalise these origins of complexity have been distinguished via an exploratory literature review.

The second knowledge question, "What types of costs are associated with the IT landscape?" shifts the focus towards the economic pillar of an organisation. Different model-based cost analyses can be used to measure monetary changes. It is important to investigate to what extent current EA models, including the application landscape, cover cost-analysis techniques. To do this an SLR was used. Furthermore, the question seeks to ascertain what the current usages of the cost of complexity are, by performing an exploratory literature review.

Table B.1 in Appendix B, showcases the sub-questions for each knowledge question and the type of literature review used for creating answering.

2.2.2 Guidelines for the Reviews

Exploratory Literature Review

As no structure is requested by an exploratory literature review, the following steps guided this research: definition of the research questions, database search (with concepts found in the RQ, such as "complexity classification" or "operationalisation of complexity"), theme and concept identification, and analysis of findings.

Systematic Literature Reviews

Different from the exploratory literature review, the SLR, used a combination of existing guidelines to ensure that the study and the findings can be validated, and replicated and that a high level of transparency can be achieved.

For the Information System field, under which EA can be placed, Webster and Watson [72] proposed the foundation for advancing knowledge by summarising a research field and finding a new research discipline. To extend their advice, Wolfswinkel et al. [73] introduced a five-stage iterative process for conducting a literature review. The authors

shifted the focus towards linking concepts discovered in academic articles rather than classifying individual papers. Instead of categorising the article itself, the most relevant ideas uncovered whilst reading the papers are grouped, leading to a more thorough review.

Hence, their proposed framework was selected and used for the SLR. To enhance the framework and provide even more clarity and transparency of the SLR, the PRISMA flow diagram [74] is applied in the "Selection" phase (to delineate the search result outcomes).

The general search strategy steps are depicted in Figure 2.1, and a more in-depth description of each stage is offered in Appendix B.

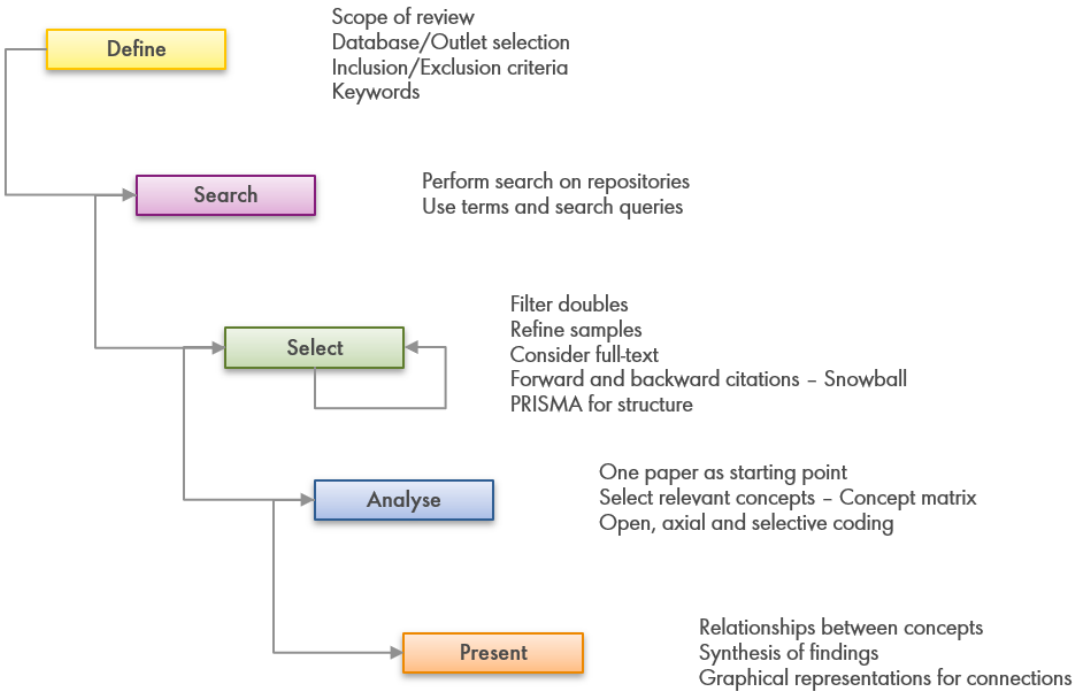


FIGURE 2.1: SLR Process Steps

2.3 Review Results

Upon understanding the search strategy used for investigating the two knowledge questions, it is now time to turn towards the findings of the literature reviews and the answers to these questions. The first part of the result section describes the insights collected whilst analysing sub-question 1, aiming at identifying the complexity metrics that can help measure architectural change. The classification of complexity is introduced, and then existing drivers of complexity are depicted, followed by existing ways to operationalise them. The second part is connected to sub-question 2, thus shifting the focus towards the economic pillar of organisations and towards understanding the costs accounted for in an IT landscape. A high-level overview of different model-based cost analysis techniques is provided. Furthermore, the current usages of the "cost of complexity" are described.

2.3.1 Complexity Drivers and Metrics

Classification of Complexity

Due to the escalating intricacy of the world, researchers in different domains started defining measurements of complexity for their respective fields of study, making it increasingly important to understand what makes a system complex and how it can be managed [45], [75].

A complex system, in literature, is defined by changes in elements that can drastically modify relationships or even a system which cannot be fully comprehended [45], [75]. Systems thinking, as a domain of study or a cognitive paradigm, defines complexity as the set of parts/components of a system that have vague or difficult-to-understand relationships which may vary across time. This field of research looks at the world in terms of wholes and relationships instead of breaking an event down into parts. This creates a paradigm shift in which a transition from an analytical perspective to a systemic one in visible.

To understand different classifications of complexity, from the myriad of existing definitions, various streams of literature have been analysed. One of the first classifications was introduced in 1948, by Weaver [49]. He proposed two views for complexity: disorganised and organised. On one hand, disorganised complexity means the existence of many variables, each having an individual erratic or unknown behaviour, as part of a system with *"orderly and analyzable average properties"* [49, p. 538]. On the other hand, organised complexity deals with *"a sizeable number of factors which are interrelated into an organic whole"* [49, p. 539].

For algorithmic and computational complexity, some of the following classifications can be considered: Kolmogorov complexity or entropy. Kolmogorov complexity is concerned with the length of the shortest computer program that produces an object as an output [50]. Entropy, in information theory, relates to the average level of uncertainty related to a variable's outcome [76]. The computational complexity of algorithms therefore touches upon the efficiency with which a problem of a certain size can be solved.

In cognitive sciences, Fioretti [51] explains that complexity should go beyond the objective view of the number of components. It should also look at the structure and behaviour of an organisation and the impact of human cognition. *"Complexity should not be seen as an objective feature of some organisational characteristics, but rather as a relative decision problem or to the representation a decision maker has of this problem"* [51, p. 499].

In social sciences, Mesjasz [77] investigates different typologies of complexities and proposes a four-type classification. The complexity categories are self-referential (e.g., logical vs. relational), gnoseological (e.g., semantic), chaotic (e.g., complex adaptive system) and computational complexity.

As the thesis is concerned with the field of Information Systems, where IT complexity refers to the number of components of an architecture, their relationships, and heterogeneity, as well as the levels of observation or subjectivity, the classification of Schneider [2] has been analysed. This was introduced in Section 1.2.2. Schneider created a multi-dimensional framework, defining complexity across four dimensions: organised vs. disorganised complexity, qualitative vs. quantitative complexity, subjective vs. objective and structural vs. dynamic complexity.

The dimensions are independent of each other, but it was shown that in practice they are intertwined and can be combined in any way. Papers [2], [20] illustrate which are the most

common combinations of the eight factors. Based on the findings, very few papers are looking at the blending of dynamic complexity (the interaction amongst agents over time with a complicated system interpretation) with disorganised systems (lots of variables and vague relationships between them)[2], [20], [78].

Sheard [5] also introduces a framework of complexity types, as seen in 2.2. For the IT landscapes, the focus, and the connection with Schneider's framework [2] is the structural and dynamic complexity (elements numbered 1-5 in 2.2). One interesting addition that this framework brings is the difference between short-term and long-term dynamic complexity. Short-term refers to sudden and unpredictable changes, whereas long-term is discussing longer time scales in which changes are planned.

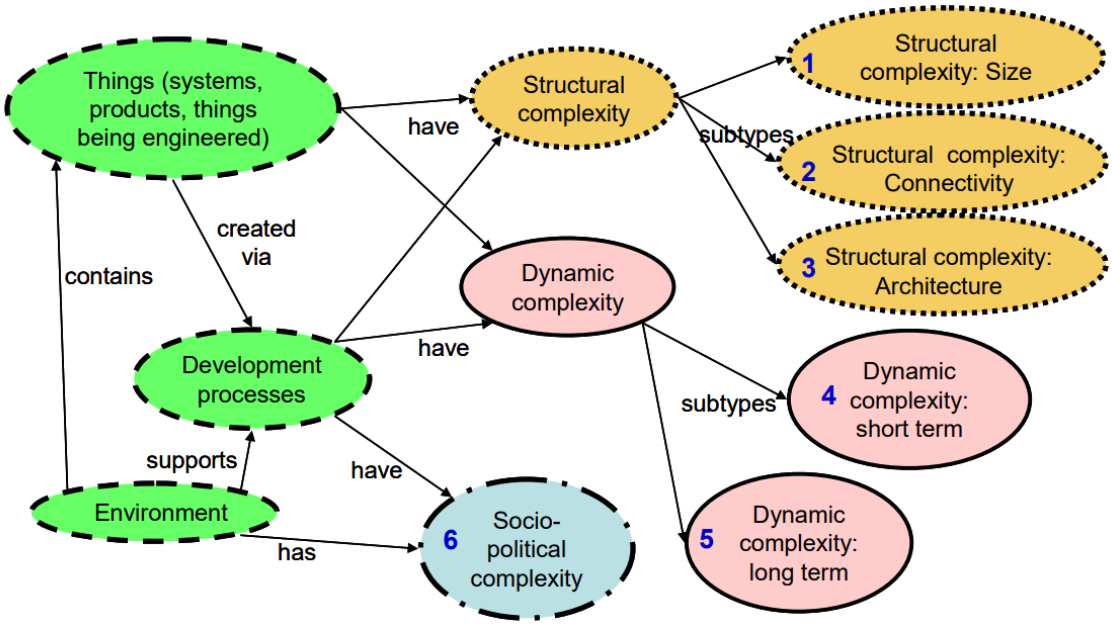


FIGURE 2.2: Relationship of Complexity Type [5]

Complexity Drivers

From the classifications and definitions described in Section 2.3.1, the focus of the thesis is linked with the architectural view on complexity. This means that complexity is related to the number of components or elements of an architecture, their relationships and the variation or heterogeneity of these. Onderdelinden et al. [18] mention that the IS architecture complexity also concerns the different levels of observation, considering the architecture as a whole or at a granular level, as well as the nature and rate of change of the components and their relationships.

Different views on factors influencing the complexity of an IT landscape were identified, with the help of the SLR.

Beese et al. [79] propose five drivers for IT architectural complexity: size, diversity, integration, planning, and dynamics. Size refers to the number of elements present in an architecture as well as the dimension of a single element (e.g., the volume of data processed by a component). The larger the number, the higher the complexity. Diversity is linked with variety (different categories), disparity (differences between categories) and balance (evenness in distribution). Integration refers to the level of interdependencies among com-

ponents, for example, task interdependency or the number of interfaces. Dynamics means the change rate of the system or the individual components. Lastly, planning is connected to humans and how aware they are of changes. One other paper introducing complexity components of IT architecture is [80]. Schüetz classifies the origin of complexity into the following categories: the number and heterogeneity of component and their relationships, the rate of change, and the application to different IT architecture domains (function, data, technology, and interfaces). Similarly to Besse et al. and Schüetz, papers [81] and [82], find the number of applications, heterogeneity, standards or functional scope as causes for IT landscape complexity.

Another aspect that seems to cause complexity is the systems integration, as described by Jain et al. [83]. An eighteen-criteria framework is proposed, depicting architectural attributes, such as interface openness or abstraction of system architecture, that impact the integration complexity.

For a more specific origin of complexity that negatively impacts costs, architecture dependability, maintenance, and evolving systems, Wehling's work [15], [84] can be analysed. The author proposed unnecessary variability as a cause for complexity.

For large-scale complex IT systems, Sommerville et al. [85] introduce two origins of complexity: trust relationships of components in a system and lack of knowledge about the system (from a human perspective). The trust relationships lead to inherent complexity, whereas the knowledge gaps produce epistemic complexity. Inherent complexity depends *"on the number, existence and nature"* [85, p. 3] of the dynamic relationships between elements in a system. The epistemic complexity is linked to the individual's lack of knowledge about a system.

Mocker [86] investigates the origins of complexity for application architecture. He discovers that interdependence (interconnectedness), redundancy (supporting the same process), the number of different applications, as well as their age impact cost (e.g., maintenance and operation), agility, and the overall IT complexity. Furthermore, the complexity of the business requirement that an application is helping with can generate overall complexity. By performing a case study at an investment bank working with 273 applications, Mocker observes that interdependency is the main origin of complexity and the only aspect that *"behaves as assumed by consultants and researchers with respect to cause and impact"* [86, p. 7]. This means that applications with more interfaces are usually the older, legacy ones, that support more processes and have higher maintenance costs. For redundancy, the case study uncovered that applications with a higher overlap do not lead to higher IT costs, but they create a complex environment.

Lentz and Bleizeffer [87] support the above-mentioned points, but they also discuss the impact of unintelligent design as a cause of IT complexity. Frequent application updates and fixes for a faster turnaround are high priorities in larger enterprises, such that everything can operate accordingly. Moreover, the authors state that users normally have specialised interaction styles with different design goals, thus not a one-size-fits-all design. These two considerations can lead to architectural technical debt. Quick fixes in one area of the enterprise can mean accepting compromises in other areas, generating a higher complexity, or cost to restore the state of the system in the future [88].

Technical debt has its roots in the software development industry. This describes the delayed technical development activities for getting short-term payoffs or timely release of specific software [89]. It means accepting compromises in a specific domain that requires

urgent attention, but overlooking how these compromises will affect other areas of an organisation. To further amplify this, it can happen that new artefacts are being built on top of such suboptimal implementations. These concerns can lead to higher costs of rework to restore the health of the system. Atchison [90] describes how technical debt and complexity go hand in hand and concludes that technical debt is the core of IT complexity.

Architectural debt resides under the umbrella of technical debt [91]. This represents the violations of code towards an intended architecture (e.g., having dependencies that cause the propagation of refactoring). Martini [91], [92] illustrates possible causes leading to architectural technical debt, for example, business factors, the use of third-party/open-source systems that were not initially part of the architecture, parallel development, non-completed refactoring (e.g., using a new application programming interface, however, the previous one cannot be removed because of backwards compatibility) or the human factor (e.g., differences in knowledge, ignorance, error-prone situations).

As technical debt and architectural technical debt are both very specific to the software domain, Hacks [88], [93] coined the term Enterprise Architecture Debt to create the link between business and IT. Enterprise Architecture Debt is a metric that depicts *"the deviation of the current present state of an enterprise from a hypothetical ideal state"*[88, p. 5]. This can mean that elements in the architecture are not implemented or executed optimally in an ideal situation. Besides elements not implemented optimally, debt is increased by bad interfaces, interoperability issues or different priorities of stakeholders. EA debt should be evaluated with respect to future evolutions, as the new developments may rest on sub-optimal implementations. To understand the taxonomy of EA debt the work of Daoudi et al. [94] serves as a reference point.

Table 2.1 has been designed to focus on classifying possible origins of complexity in the five groups proposed by Beese [79]. The elements which could not fit into the categories are part of the "Other" group.

Table 2.1: Causes of Complexity in an IT Landscape

Category	Indicator	Meaning
Diversity	Unnecessary variability (Similarity)	Alternatives (to an artifact) exist; thus, the artifact is not required to serve the architecture.
	Functional scope	The number of business functions realised by the application.
	Task interdependency/ Redundancy	The number of components that support the same processes; the degree to which an application covers certain functionality already covered by another application.
Size	Different applications/elements	The number of different elements in an architecture.
	Dimension of a single element	e.g., The volume of data processed by a component

Continued on next page

Table 2.1: Causes of Complexity in an IT Landscape (Continued)

Category	Indicator	Meaning
	Infrastructure components	The number of infrastructure components realising an application.
Integration	Application interfaces/information flows	The number of application interfaces/information flows/integrations that compose the landscape.
	Interconnect or Integration patterns	Width of integration and integration pattern complexity.
	Coupling	Loose or tight couplings between applications.
Dynamics	Change rate in number and variety of components	Frequency of changes in the architecture.
	Trust relationships (Inherent complexity)	If one component (A) does not trust another one (B), but after checks it can trust it.
Planning	Lack of knowledge about the system (Epistemic complexity)	The architect does not know about the system's components and relationships.
Other	Communication standards	Applications may use different communication standards which can lead to complexity.
	Age/Legacy system	The age of the system can lead to complexity. Over time many changes get implemented in a rush rather than a well-designed way, this means that older applications could be more complex and less frequently exposed to changes.
	Standard conformity	Check the type of application: buy (available on the market), make/buy (buy and make alterations), customise (full customisation and build of the application).
	Standardisation of constructing elements of the component, infrastructure, technology stack	The standardisation of the constructing element infrastructure. What the application is built on follows the same standards.
	Architectural debt	Accepting compromises in one dimension to meet another urgent demand in another dimension.

Continued on next page

Table 2.1: Causes of Complexity in an IT Landscape (Continued)

Category	Indicator	Meaning
	Enterprise Architecture Debt	Deviation of the current present state of an enterprise from a hypothetical ideal state.

Operationalisation of Complexity

Quantifying architectural complexity is a difficult task, as a myriad of factors, as depicted above, can impact the overall architecture. Saat mentions that elements and relationships of an architecture change over time, hence challenges of complexity measurement are also concerned with dynamic aspects such as: *"the volatility of architectural layers, architectural elements and their interrelationships (frequencies of change and lifecycles)"* or the *"prediction and management of impacts of changes on different granularity levels"*[95, p. 3].

The *"lack of any generally accepted methodology for complexity measurements"* [20, p. 115] has been identified as a challenge in the past [96]. To address this gap, Iacob et al. [20] performed a systematic literature review (assessing 20 prevalent papers) to make an inventory of complexity metrics used in different domains. 42 complexity metrics, as seen in Appendix C were identified. Some of the identified metrics are the number of relations, number of elements, cyclomatic complexity, conformity, or redundancy.

Based on this classification and the work of Iacob et al. [20] it is visible that the dynamic aspect of complexity is not addressed in the literature.

To build upon the work performed by Iacob et al. [20], the academic works which were written after the publication of their research and which can bring additional insights into the operationalisation of complexity have been inspected.

One recent development in the literature, focused more on the dynamic aspects of complexity is Adaptive Enterprise Architecture. Considering that adaptation is essential in highly competitive and dynamic business environments, Daoudi [66] introduces this approach of proactive sensing and responding to change, to explicitly manage trade-offs. In moving from as-is to to-be scenarios, the EA Degree of Dynamic Complexity (DDC) is proposed, as seen in equation 2.1. i is the indicator of the EA version (i is the current architecture, i+1 is a possible to-be architecture), the first element of the sum is concerned with the values of dynamic factors (changing in time), whereas the second one is related to static factors. N represents the number of factors selected for the analysis.

$$\text{DDC i, i+1(t)} = \sum_{j=1}^{n} \frac{fj(t)+fj}{n}, \text{ where } n \in N \tag{2.1}$$

Furthermore, paper [66] lists existing types of calculation for some complexity metrics, such as an interdependence matrix (for measuring layer interdependency Business-Application-Technology), entropy (for heterogeneity) or a context awareness capability matrix (for subjective complexity).

Another important complexity measurement that can be used to choose between EA alternatives is proposed by Rojas [97]. He mentions that complexity metrics at the implementation stage identify elements that can either reduce or increase complexity, for example,

a high cohesion between elements, respectively a high coupling between elements. For the design phase, to re-estimate project effort and understand the complexity of EAs, Rojas proposes the Structural Complexity Measure (SCM), as seen in equation 2.2. This measure looks at functionalities and dependencies at different architectural levels, for example checking the dependencies between projects (level 0) or between processes, applications, and data (level 1).

$$\text{S C M}=\text{F}^{3.11} + D^{3.11} \tag{2.2}$$

To combine the work of Iacob et al. [20] with the new developments in the field, table C.1, in Appendix C has been designed. The concepts present in the table are selected from relevant contributions in the field which operationalise complexity factors.

Answer to RQ 1

The first sub-RQ, "Which complexity metrics can be used to measure change in an IT landscape?" was aimed at identifying metrics that can help with the quantification of complexity during change processes.

The literature on the complexity of IT is very dispersed, with each author trying to either zoom in on a specific aspect, such as application complexity or offer a high-level image of what can be considered a factor or metric for complexity. No study offers a clear understanding of what are the causes and the possible means to measure complexity at a holistic and dynamic IT landscape level.

For the complexity origins, several factors are highlighted in the literature. By leveraging the empirically tested hypotheses of Beese [79] showing the existence of relationships between drivers of complexity, and the strong effect of structural complexity on dynamic complexity, all these factors leading to complexity can be experienced in the transformation/change of the IT landscape.

When performing a search for identifying complexity metrics, it was rarely the case that dynamic aspects such as volatility, and interdependencies of impact of change were addressed. EA Degree of Dynamic Complexity [66] was one metric trying to address the dynamic aspect of complexity.

Table 2.2 has been designed as the answer for the first sub-question. Hence, the metrics in the table represent the prevalent findings of the exploratory and systematic literature reviews.

In the table, there are three levels of the IT landscape (business – application - technology), as supported by ArchiMate's core metamodel [1] seen in Figure 1.1, such that complexity origins and metrics can be linked upwards with the business and downwards to the technology layer. The "Other" category in the table, contains complexity drivers and metrics that can be applied to and influence the whole architectural landscape.

As the scope of the thesis is limited to the application landscape complexity, the drivers and indicators in this category are selected for the case study.

Table 2.2: IT Landscape Complexity Metrics

Level	Indicator	Meaning	Metric
Business	Unnecessary variability	Alternatives (to an artefact) exist; thus, the artefact is not required to serve the architecture.	Create an influence relationship matrix or design structure matrix.
	Functional scope	The number of business functions realised by the application.	Count the number of business functions that an application realises.
	Task interdependence or Redundancy	The number of components that support the same processes.	Create an influence relationship matrix or design structure matrix.
Application	Number of application-s/elements	The number of different elements in an architecture.	Count the total number of elements.
	Dimension of application	e.g., The volume of data processed by a component.	Verify the size of the component.
	Interfaces	The number of application interfaces/information flows/integrations that compose the landscape.	Count the total number of relations. Use SCM for functionalities and dependencies. Use structural complexity formula (based on number and heterogeneity).
	Heterogeneity	Entropy of element or relation.	Use the entropy formula for either relationships or components; e.g., verify functionality overlaps or data overlaps
	Standard communication	Applications may use different communication standards which can lead to complexity.	Verify the communication standard of components.
	Standard conformity	Check the type of application: buy (available on the market), make/buy (buy and make alterations), customise (full customisation and build of the application).	Application type complexity – number and heterogeneity of the customisation levels (make, buy and customise or buy of a domain application).

Continued on next page

Table 2.2: IT Landscape Complexity Metrics (Continued)

Level	Indicator	Meaning	Metric
	Age	The age of the system can lead to complexity. Over time many changes get implemented in a rush rather than a well-designed way, this means that older applications could be more complex and less frequently exposed to changes.	Verify the age of the application.
	Interconnect	Width of integration and integration pattern complexity.	Use cyclomatic complexity for architecture.
	Coupling	Loose or tight couplings between applications.	Use the coupling factor.
Technology	Components	Different types of technology realise an application.	Count the different types of technology (e.g., number of OS, middleware, databases).
	Standards for infrastructure	The standardisation of the constructing element infrastructure. What the application is built on follows the same standards.	Verify the used enterprise standards.
Other/Dyn.	Change rate	Frequency of architectural changes.	Check the number of changes in a time period.
	To-Be architecture	Choose a to-be architecture.	Use Fuzzy AHP.
	Trust	If one component does not trust B, but later after checking it trusts it.	Use Degree of Dynamic Complexity.
	Lack of knowledge	The architect does not know about the system's components and relationships.	Check the knowledge of the architect in a subjective manner. Using an awareness matrix can help.
	EA debt	Various EA smells metrics can be selected.	Use the online catalogue created by Hacks [98] and apply it to ArchiMate models.

Continued on next page

Table 2.2: IT Landscape Complexity Metrics (Continued)

Level	Indicator	Meaning	Metric
	Propagation cost	The cost incurred when a randomly chosen application is changed.	Use visibility fan-in and out, as well as the propagation cost.

2.3.2 Costs of an IT Landscape

Analysing costs is an essential aspect of every enterprise, and cost concerns are one of the main stakeholders' interests [99]. This section aims to provide an answer to the second sub-RQ, "Which types of costs are associated with an IT landscape?", by shifting the focus towards the economic pillar of an organisation.

As the quantification of complexity should be done based on designs depicting the IT landscape, it is first investigated what types of costs normally emerge when performing model-based cost analysis. Next, the meaning of the cost of complexity is researched. By looking at both categories (costs accounted for based on designs and the costs linked with complexity), several IT costs emerged.

Model-based IT Cost Analysis

As discussed in Section 1.1, architectural representations are expected to help an enterprise manage complexity. However, using EA models to quantify different types of costs is an aspect which is mostly overlooked in both academia and practice. Miguens acknowledges the fact that EA misses a viewpoint for representing costs and mentions that there is a "lack of solutions to integrate EA models with cost models" [99, p. 484].

Cost Associated with ArchiMate Components

Currently, most organisations use TOGAF as a framework for building architectures. One of the modelling languages supporting this framework, used worldwide by practitioners and scholars is ArchiMate [100]. The core model of ArchiMate allows extensions by adding various attributes to concepts and relationships [99],[101]. Several papers addressing how ArchiMate models can help with cost analysis are further discussed, as well as other model-based approaches for depicting costs.

For performing quantitative analysis of EA models (especially with ArchiMate), previous works introduce approaches to quantify workload by measuring effort through performance measures or costs [56], [57]. Given an ArchiMate model, a top-down and a bottom-up method can be described. Starting with the top-down approach, this means that the higher layers in a model impose a workload on the lower ones, whereas the performance and cost of lower layers impact the higher ones. As cost concepts are not part of ArchiMate, the authors of the papers mention that attributes can be added to components to quantify concepts and relationships in such models. Hence, if a design is created in ArchiMate and one adds the license cost as an attribute for an application component, this is permitted by the modelling language. To then perform a cost analysis, the propagation of clearly defined costs through the layers can be observed.

Several types of IT costs can be depicted in a model. These depend on the different stakeholder's perspectives (the level of observation). For example, the user of a solution is concerned with the response time (e.g., price per use, the cost of a service/product),

from a process perspective the completion time is important (e.g., the sum of all resource completion costs), the product is connected with the processing time (e.g., the sum of all business resources completion cost), the system is linked with throughput and the resources that must be utilised (e.g., they will have a variable or a fixed cost assigned to them).

By using a model-based approach an architect can arrive at better IT investment decisions considering project costs, risks or benefits of a certain solution [102]. Once again, the work done by Aldea et al. [102] supports the fact that cost can be an attribute of any architectural element. For IT project selection, cost is seen as the "cost of a project", and it must represent the quantity of a resource needed for an activity [101]. Other types of costs, mentioned by Iacob et al. [101], that can be part of the IT landscape, and specifically in the application landscape are application component costs or interface costs.

Based on the work presented above and the consensus that costs can be attributed to elements in architectural designs (e.g., in ArchiMate) and propagated from one element to the other, Miguens [99] created a viewpoint for costs in ArchiMate. This helps with calculating the cost of services or products, by combining time-based activity costing (measuring capacity cost rate, cost of activity, time taken to perform an activity) and business processes. Several types of costs are assumed to encompass the "final value" (e.g., cost of a service). These are the capacity cost rate (cost of resource divided by the amount of time that resource is available to work), the cost of capacity supplied or the cost of an activity (time taken multiplied by a unit cost).

Välja [103] exemplifies the analysis of costs, alongside availability and interoperability, with an ArchiMate-based example. In the case of microgrid controllers, the characteristics of system architecture containing such embedded systems, are depicted. Each concept in ArchiMate is illustrated as a class which can have multiple attributes, and costs are some of them. The following concepts were assigned initial and yearly costs: business service, business function, application service, function, component, and infrastructure service or component.

As the costs can be illustrated as attributes for elements in ArchiMate, another prevalent work [104] introduces a framework for IT investment cost assessment. The cost taxonomy, which includes elements such as project management costs or operation and maintenance costs, can guide an architect in quantifying IT landscape costs. One important element stated in the paper is that most IT investment appraisals are focused on the most obvious costs [104]. The work of the authors also describes important drivers of costs, such as geographical spread which can influence both the cost and the level of complexity of the business process re-engineering, the scenario complexity, the number of systems, standards, or the team's ability to implement a change.

Other Costs Applied to Models

Moving from ArchiMate-specific cost analysis, Lagerström [105],[106] introduced a metamodel for modifiability analysis. The author mentions that most business processes are supported by software systems which need to be modified (e.g., either extended, deleted, adapted, or restructured) to keep up with the dynamic business environment. These modifications can range from a few functional requirements in a single system, to creating a SOA for the whole enterprise [105]. The cost of change in the IT landscape is measured in the number of man-hours needed to implement the modifications. This is directly influenced by the expertise of the architects/developers and the number of architects, the maturity of a change management process, the synchronization needs, as well as the change difficulty

or size.

Paper [107] provides a method for assessing organisational decisions in terms of cost and profit. The cost-benefit analysis method is used for measuring costs and to scale the return on investment (ROI) of different EA scenarios. Based on different business goals (EA scenarios), different activities must be performed, and given the rankings of quality attributes, such as flexibility, efficiency or security, the ROI will be equal to the benefits calculated divided by the total cost of a project estimated by experts.

Li [108] describes a semantic eXtensible Business Reporting Language model for enterprise total cost analysis. To understand different types of costs, an ontology for financial terms is used. The two types of costs discussed in the paper are activity and product costs. The activity cost is linked with the number of expenses, and the product cost is the general management expense.

Cost of Complexity

Transitioning to what the combination of "cost of complexity" can mean, the literature rarely tries to investigate what are invisible costs that are driven by a complex IT landscape. In most cases, the complexity of an IT landscape is associated with a higher total cost of ownership (TCO). This stems from looking at only the number of applications or infrastructure components and quantifying the expenses spent on purchasing, running and maintaining them. Not enough attention is given to what other types of costs can be experienced by an organisation due to the complexity of an IT landscape.

Sometimes, authors such as Swenson [109] mention that the cost of complexity is linked with the overhead costs, the percentage of total product cost and the percentage of revenues. Other times, looking at costs and complexity is defined by using a part number count. By doing this, one can measure the cost of introducing or maintaining a new component in the system. However, the cost of complexity does not grow in proportion to the number of components in the IT landscape, but in proportion to the number of links between them as they interact with one another [110].

Thus, only by saying that a new part increases the complexity of the landscape and the costs associated with the expenses incurred for the new part grow, is not sufficient. It is also important to note that generally, costs of individual items are considered (e.g., application costs, license costs etc.), but the cost of complexity is highly dependent on the component that was previously in place or new links between components [110], [111].

Several papers touching upon the connection between cost and complexity have been analysed and are described below.

Costa [112] describes approaches to quantify complexity, such as entropy, the Kolmogorov Complexity or dynamic system complexity, and the costs emerging due to complexity. In his work, complexity is based on costs related to mapping entities into models or errors in reconstructing the representations of the models. Thus, the complexity is related to the development of a complete and accurate model and the error of reconstruction [112]. If these two types of costs: modelling (how much one invests in representing the state of the system) and restructuring are high, then complexity also increases. Nonetheless, the more accurate the model is, the less costs of reconstructing the model will be incurred. Thus, costs are related to complexity as the following formula holds: complexity is equal to the cost of modelling plus the cost of restructuring. This formula is influenced by the person looking at a model and the available resources for modelling.

Another prevalent work, [113], aimed at presenting techniques for estimating the cost of complexity and bridging the gap concerned with the lack of quantitative work helping managers and designers understand this cost. Sturtevant [113] only mentions, from a complexity prism, that highly complex software impacts the productivity of employees (impaired by complexity), the turnover (people leaving due to architectural complexity), and defects (having a higher density in architectural complex source codes).To control complexity, hierarchy, modularity, and abstraction layering can be employed (e.g., use design structure matrix and group applications into "core" or "peripheral").

The work of Diao et al. [114] connects with the two above-mentioned papers as it showcases that complexity metrics must be related to a model. Evaluations can then be performed. The relationship between complexity scores and the time it takes an individual to act will lead to a quantitative measure of costs. To manage complexity, standardised components are preferred instead of custom-built ones.

For software development projects, two papers address the cost and complexity relationship. The first one, [115] suggests that 16 complexity factors must be rated first and then by summing them (using parameters and weighting) the cost will be expressed in man-hours times the effort. The second paper, [116], represents a proposal in which the importance of determining the cost of change is emphasised. The author hypothesises that *"the cost of change is directly related to the structure of the design of a software system and the metrics proposed"* [116, p. 31].

In the field of product development and engineering, the work of Georgiades et al. [117] describes the cost propagation due to design changes. The authors create a methodology for predicting, visualising, and assessing the propagation of change and the cost associated with it. This method is called "Cost Prediction for Change Propagation" or CP2. Based on the dependencies of components and expert opinions on how these influence each other and what costs can be incurred, a correct calculation of the cost of change will be revealed. This preference of assigning the costs by experts is also present in paper [118]. The authors propose a heuristic-based complexity method to estimate costs. It is described as applicable to all systems. Based on expert opinions cost-inducing parameters are assigned and complexity magnitudes are given.

Lastly, an interesting approach for turning economic value from subjective into objective with the help of complexity theory is introduced by Murialdo and Cifuentes [119]. As also explained by Sturtevant [113], to understand how to measure effective complexity one must break down the ensemble and look at its regularities. This is additional to embodied information which can be measured with, for example, entropy. Hence, the effective complexity goes beyond intrinsic value and considers how in the future an object (physical, idea, service, application etc.) can lead to an overall higher complexity. There are two valuations discussed in the paper: the total complexity (TC) and the complexity opportunity cost (COC). To determine TC, the object of interest and its context/surroundings are encoded in a simulation as information. There will be two simulations, one with the object and its context and one only with the context/surroundings (making abstraction of the object), run for a period of time. The difference between these two results will be the difference in complexity between the two worlds.

COC is equal to the total complexity added to the system if the time and resources had not been expended elsewhere. For example, the resources which are present at hand mean less complexity than the ones that should be acquired/updated etc. The value, in terms of price and compensation, will be the intersection of total complexity and the COC (loss

without the individual object).

Answer to RQ 2

The second sub-RQ "What types of costs are associated with IT landscapes?" was formulated to create the design of a comprehensive overview of possible costs connected to IT landscapes.

The systematic literature review performed for sub-RQ 2a showcased the lack of cost models applied to EA models, as techniques for the quantitative analysis of architectural models hardly exist [56]. Furthermore, the existing literature mentions determining visible costs such as project, product, and service costs but only alludes to the costs that can be experienced due to a complex IT landscape.

As sub-RQ 2a aimed at identifying best practices for model-based cost analysis, a few ideas must be acknowledged. It is important to note that in ArchiMate any type of cost can be attributed to an element and propagated via a bottom-up approach. This ultimately can allow a summation of costs for the total landscape.

Model-based cost analysis also implies using financial terms and taxonomies or ontologies for IT landscape costs. Now, for most of the important decisions taken in companies, the guiding costs are the "obvious ones" [104], the direct or the visible costs. The majority of costs used in taxonomies and later represented in models fall under the investment or operation and maintenance categories, and it is rarely the case that costs incurred during change processes or due to a complex landscape are described.

This underlying idea is also supported by considering the comparison of cost taxonomies for information systems management, introduced by Irani et al. [120]. Some of the least discussed costs are part of the change or project management categories [120]. Moreover, considering frameworks for IT governance, such as the Information Technology Infrastructure Library (ITIL), this focus on linking the operational costs with the complexity of the IT infrastructure is highlighted once more [121].

Sub-RQ 2b tried to uncover the use of "cost of complexity" in different domains. It was observed the common approach in the synthesis of terms cost and complexity is to first determine the costs of individual parts and look at an element's intrinsic value, rather than consider the components as highly dependent on each other and see how landscape complexity can drive costs.

The cost of complexity can then be said to grow in proportion to the number of links and interactions of components over time. It is important to also note the level of abstraction of the IT landscape, as sometimes modelling an IT architecture more thoroughly can potentially decrease the cost of restructuring.

One recurring element mentioned was that the cost of complexity must be based on simple measurements. *"Often humans treasure things for their simplicity"* [119, p.75] and by breaking down the cost of complexity into more manageable parts, a more objective view can be created.

Once these components leading to complexity are identified, different calculations should be in place, for example, calculating the probability of a design change, model rework or the product cost. The costs are mostly assigned by experts (e.g., by giving weights). This means that the cost of complexity can be calculated as a mathematical formula between the complexity factor score and the expert-assigned costs.

In Figure 2.3, an overview of IT landscape costs is introduced. The costs are assigned into two categories visible or hidden costs. The visible costs are the direct ones, normally accounted for when working with IT (pieced together during the literature reviews), whereas the "hidden ones" are the costs that can emerge when IT landscape complexity hinders the change process of an organisation.

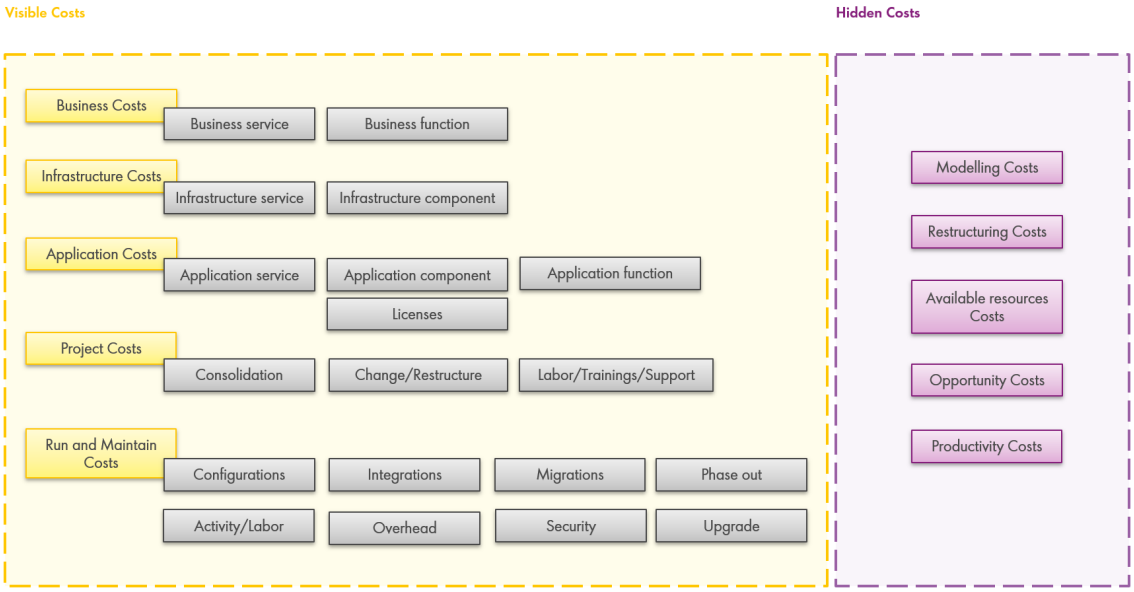


FIGURE 2.3: IT Landscape Costs

2.4 Summary and Takeaways

The chapter starts by introducing the two knowledge questions that try to ascertain what complexity metrics can be used to measure changes in an IT landscape, as well as create an overview of types of costs associated with complexity. To answer these questions, once they have been defined, a search of prevalent literature was performed, followed by a selection and analysis of results which were ultimately synthesised and presented in this chapter.

The key takeaways from Chapter 2 are the following:

- There are several ways to classify complexity, but for IT landscape complexity, where one needs to consider the number of components of an architecture, the relationships, their heterogeneity and the levels of observation, the classification of Schneider [2] is appropriate.
- Both short-term and long-term dynamic complexity are under-investigated areas of research.
- Numerous drivers for IT landscape complexity are present in the literature, where the most-mentioned are: size, age, diversity, unintelligent design leading to technical debt, and integration/interconnectedness.
- Complexity metrics range from simple counters, such as the number of relationships, and elements to cyclomatic complexity or entropy. However, when dealing with dynamics and changes in the application landscape the EA DDC, SCM and Fuzzy AHP (as formulas) can be employed.

- When performing model-based cost analysis or when considering changing the IT landscape, the majority of costs accounted for fall under the total cost of ownership category. This means acknowledging project, application licenses, interfaces, and run, operate and maintain type of costs.
- Based on the literature study, it was identified that not enough attention is given to the invisible costs (of change) emerging due to a complex IT landscape. Furthermore, the cost of complexity is expected to be directly proportional to the number of interconnections between elements and their interaction.

Chapter 3

Problem Investigation

Chapter three proceeds to showcase what methods guided the formation of an answer to the design questions (RQs 3-5), introduced in Section 1.4.1. Having identified complexity drivers, metrics and costs associated with the IT landscape (as seen in Chapter 2) and starting from the premise that complexity hinders the overall understanding of a landscape when a change is made, the inputs for creating the artefact are collected through a case study.

Section 3.1 begins with a description of what performing a case study entails. Moreover, an introduction to the enterprise at which the thesis was conducted is given. The next section, 3.2, describes the goals, outcomes and the overall structure of the survey performed as part of the case study. The same elements are then detailed in Section 3.3 for the two types of interviews conducted during the research: one concerned with the complexity of an application landscape and one on its costs. This chapter showcases the results from the information retrieval procedures and their analysis. This can be seen in Section 3.4. To conclude, the key takeaways from this chapter can be found in Section 3.5.

3.1 Case Study Design

In this section, the case study methodology is detailed. Next, the enterprise at which the case study has been conducted is introduced. Furthermore, the connections between the literature findings and the current context of the company are depicted. Lastly, a detailed explanation of the stages guiding the case study within the company is presented.

3.1.1 Phases of Case Study

A case study represents a qualitative research method. This is an empirical inquiry, suitable for investigating a phenomenon in a real-life setting [122]. Moreover, patterns are visible in an easier manner after performing observations in the field.

According to Yin [122], case study research involves 6 steps: plan, design, prepare, collect, analyse and share. In the plan phase, a case study is selected based on research questions that aim to investigate a contemporary phenomenon. Then, during the design stage, a hypothesis is defined, alongside validity or reliability criteria. In the preparation step, the researcher must revise their skills and create a thorough study protocol. For the collect phase, different data is gathered and promising patterns, insights and concepts become visible. The case study evidence can be collected from primary (e.g., interviews,

questionnaires) or secondary data sources (e.g., documentation). The analyse stage is concerned with creating links between the findings and drawing conclusions that can be shared with the intended audience during the Share phase.

Following the approach explained by Yin, Tellis [123] proposed an application of a case study methodology. By having four stages (which encompass the six phases introduced by Yin [122]): design the case study (skills needed, protocol), conduct the case study (data collection, questionnaire, interviews), analyse evidence (analytic strategy) and develop conclusions, recommendations and implications, interesting findings can come to light.

The thesis, therefore, follows these four steps, as seen in Figure 3.1.

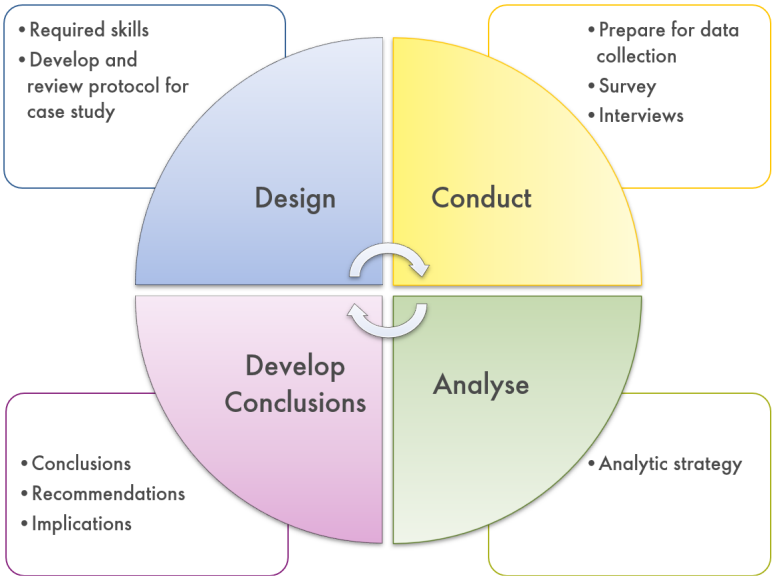


FIGURE 3.1: Case Study Phases

3.1.2 Company Description

To understand how complexity can be managed and quantified, a case study approach has been selected.

Overview of the Enterprise

For the research, the case study was conducted within a large organisation focused on providing energy solutions to customers all over the globe. The company operates in more than 70 countries worldwide and some of its core activities include the exploration, production, refining or marketing of oil and gas products. Additionally, alternative or renewable energy solutions are part of the activities that the organisation is concerned with. The energy solutions, thus range from fuels and chemicals to hydrogen or liquefied natural gas.

Due to this global spread and the diversified business activities, the company is structured in numerous teams and business units, having a large number of employees that work in refineries, production facilities and also the corporate environment.

When talking about the IT landscape of this company, currently, the number of applications used by the organisation is in the range of thousands, supporting a diversified portfolio.

Furthermore, advanced technologies are being used to help build a more sustainable future and serve the needs of both commercial and industrial customers daily. Size-wise, this creates a large IT landscape.

To remain competitive in the ever-changing and evolving environment, the organisation needs to generate value from their IT landscape. The architectural department of the organisation recognises the need to simplify and integrate the platforms and architecture roadmaps, having the right resources (both technical and human) to navigate an intricate IT landscape, whilst focusing on cost and value delivery. This department helps with various pillars of the organisation such as asset management, human resources or legal.

Problem in Context

Currently, this large enterprise aims at simplification and integration of various application components or technologies. Simplification is connected to the process of rationalising the IT landscape. On one hand, this means, the organisation wants to get more visibility about the components of the landscape to reduce the costs required for applications or technologies. On the other hand, this higher visibility means a better decision-making process. With informed decisions, an agile roadmap towards the future is expected to be of higher quality. By having this in place, more customer value and alignment between business and IT can be achieved. However, changing quickly to meet business requirements (having the necessary agility and flexibility), is inhibited due to the complexity of the IT landscape.

Numerous initiatives and guidelines exist that touch upon the simplicity of an IT landscape, however, none provide a method for visualising the value of complexity or the costs associated with the change process in a complex environment. Hence, it is important to observe what complexity indicators are relevant for the enterprise, what drives them, and how a combination of these can quantify complexity. It should also be investigated what types of costs are usually driven by the complexity of the IT landscape.

As mentioned in the introduction, from the overarching IT landscape, a smaller subset is selected as the scope of analysis in the case study. This is the application landscape.

Therefore, application landscape complexity (considering the application components, their interconnections, and the data exchanged) must be calculated. To be able to quantify this type of complexity, a limited scope within an enterprise has been selected. This is due to the fact that quantifying ALC for the complete enterprise is a far too broad research area.

The decision was thus made to focus on quantifying ALC for a limited scope - the changes made by projects. The belief is that quantifying complexity in such a manner helps with selecting the best options in projects.

Therefore, for projects, the type that can assist in creating a method for quantifying ALC is the one which includes detailed solution options. This means having a representation of the current state in place, alongside the different future options. These documented solutions help in observing how architects perceive the complexity of the application landscape, and whether or not complexity hinders the agility of an enterprise.

3.1.3 Case Study Design

In a real-life setting, it is important to see if the literature findings discussed in Chapter 2 still hold. Once mapping the literature to the organisation and uncovering requirements

and new insights, the artefact design can be guided with the help of the four steps defined by Tellis [123].

Design

The first stage is the development of the case study protocol. Two subsequent steps must be in place: determining the required skills of the researcher and reviewing the protocol. According to Yin [122], the skills needed to engage in a case study range from having the ability to ask good questions and interpret responses to having a firm grasp on the issue studied. For the protocol, this means creating a plan of approach with key stakeholders. For the thesis, several meetings were conducted with the company and university supervisors to ensure that the study design was achievable and sound.

During the case study, the involved participants were split into two groups. The first is the advisory group, formed by four experienced architects working in different parts of the enterprise. This group was regularly informed of the research progress, provided advice and assisted in the validation of the case study results. Next, is the "interview group" composed of IT managers, lead architects, solution architects and data architects, as well as strategy and planning representatives. Furthermore, additional discussions were conducted with people having various functions in the organisation to understand how the topic of application landscape complexity and its associated costs is interpreted.

Conduct

The second stage is "Conducting the case study". To conduct the case study, both a survey and interviews were performed, as these helped with the creation of a sound study of the investigated phenomena [68]. Furthermore, these two approaches are practical ways which allow the inclusion of personal experience in the "construction of knowledge" [124]. Hence, the choice to conduct the case study via interviews and a survey is justified by the desire to create an in-depth understanding of the researched topic by combining multiple sources of evidence.

Additionally, by involving participants in a survey and interview, it is believed that the designed artefact in this research will have a better acceptance once proposed.

Three steps must be performed to carry out a successful research project. The first is preparation for data collection. A study database is constructed and is comprised of any observations, consulted documents, additional literature, and remarks from open-ended conversations.

The next step is the distribution of a questionnaire/survey. This is described in detail in Section 3.2. Lastly, interviews need to be conducted, for a better focus on the case study topics. Section 3.3 explains the two types of interviews used in this case study. For both the survey and the interviews, numerous ethical principles were followed based on the extensive list presented by Oldendick [125]. Furthermore, the guidelines used by the Ethics Committee of the University of Twente were followed. Some of these principles guiding the research are the voluntary participation of the people involved in the research, informed consent, anonymity and data protection. As transparency was an essential factor in this research from the beginning, the participants in the case study were regularly updated with the collected insights.

Different sources of evidence were consulted. These are the survey results, the insights from the interviews or the observations made by conversations in the office. According to

Yin [122], by having multiple sources or achieving the triangulation of evidence, creating a study database and maintaining a chain of evidence, the reliability of the study increases.

Analyse

The third stage of the case study is the analytic strategy development. Sections 3.4.1 and 3.4.2 explain the undertaken analysis process for the survey and the interviews.

Conclude

Lastly, the most important aspect from a user perspective is the presentation of answers. To use the uncovered findings further, the thesis must use appropriate language, aligned with the one used within the organisation. Besides the conclusions, recommendations on how to develop the artefact need to be provided.

3.2 Survey

This section first explains the reasons for using a survey as a means of collecting data during the research. Next, the process underpinning the survey is detailed.

3.2.1 Goals of Survey

To design a measurement method for quantifying ALC for an organisation, it is important to start by understanding its current positioning in terms of complexity management. Moreover, the perception of stakeholders on complexity drivers needs to be investigated. A survey has been distributed within the organisation, to familiarise participants with the research. As the case study uses both a survey and interviews, the participants will be made aware of the topic of the thesis before the interview. The survey helps in gathering initial opinions on the topic of application landscape complexity.

A survey is *"a method of gathering information from a sample of individuals"* [126, p. 9]. The survey allows participants to respond to questions at their preferred time [127]. A combination of open-ended, multiple-choice, closed and rank-order questions is selected. This keeps the participant engaged and enhances the reliability of the data collected.

To ensure that both business and IT perspectives will be fused with the help of the survey, the intended audience was defined as IT managers, solution and lead architects, as well as data architects. Other perspectives going beyond this intended audience were welcomed if the participants had knowledge of the IT field. This sampling is called "purposeful sampling" given that the selected participants match the criteria of the research questions and they can be considered experts in the phenomenon being studied [128]. Once sent out, 32 answers have been received, from a pool of 38 desired participants.

For the thesis, the following outcomes were sought: understanding if the application landscape of the organisation is perceived as complex and what type of connotation (positive or negative) is associated with complexity, gathering drivers of complexity, ranking literature-identified complexity indicators, as well as discovering sources of information, tools or methods that could potentially be used for quantifying complexity within the enterprise.

One important aspect that needs to be explained is how the selection of the indicators in the ranking of complexity was performed. Based on the initial list of complexity drivers and indicators, seen in table 2.2, an expert (the company supervisor) was asked to make

a selection of these given their occurrence in the company throughout the years. This practice aids with the validity of the survey [129].

It is expected that the survey will shed light on how complexity is perceived, for example, if it is a challenge or not, or what capabilities for complexity management exist in the organisation. Furthermore, the drivers of complexity and the rankings of literature metrics can indicate what the initial focus of the measurement method should be. The survey is also discovering what information sources, from within the enterprise, can be used in the measurement method for creating a clear and accurate image of complexity.

3.2.2 Process for Conducting the Survey

Data Collection

The survey was created using Microsoft Forms (MSForms), as this represents the standard for the organisation at which the case study was performed. From a confidentiality standpoint, the survey only collected the names of the respondents for the mere purpose of inviting them for follow-up interviews. An opening statement has been composed which offered an overview of the survey's purpose.

Before sending the survey to a broader audience, three rounds of improvements were in place. This is commonly known as "pretesting" [127]. First, one of the solution architects went through the questions to make sure that they were understandable and in line with the language used within the organisation. Then, one lead architect, the company supervisor for the thesis, suggested changes for gaining more concrete and tailored answers that could help achieve the expectations stated in Section 3.2.1. The last round, "the mock survey", implied handing out the survey via email to one of the employees who were part of the target audience, analysing the responses and then discussing how the questions could be reshaped to tap into even more specific knowledge of participants.

By having such a thorough pretest approach, the questions perceived as difficult to understand or the ones interpreted differently than intended were corrected. The questions of the final version of the survey can be seen in Appendix D.1.

Data Analysis

The data collected was analysed by using a sequential approach. The data was grouped under 4 main categories: introduction to complexity, maturity of application landscape complexity management, complexity factors and costs, and final remarks that could indicate the need for a measurement method. This allowed a structured way of analysing the responses.

First, the holistic results of each section were analysed by employing the outcomes generated by MSForms, and later the responses were compared by looking at individual answers [130]. Moreover, links between the 4 categories were connected and a thematic analysis was performed, as some questions were open-ended [131]. The raw data collected from the form supported the analysis, as recommended by Ball [132].

As the purpose of the survey is to create a basis for answering the RQs, as well as to partition the responses according to group characteristics [132], the outcomes of the analysis are found and discussed in 3.4.1.

Data Quality

To ensure data quality, the topics of reliability and validity were acknowledged in the survey design.

Starting with validity, in online surveys, a big threat is the change of the questions *"between time periods and administration"* [133, p. 3]. During the set time for completion, the survey questions were not changed. Furthermore, once the survey was closed, the respondents were sent the outcomes generated by MSForms and during the later stage of the research, in interviews, they were asked if they agreed with these. The outcomes of the analysis were also discussed with the advisory board, for data triangulation. Other factors such as face, content, and internal or external validity were checked with the help of the pretests of the survey.

Moving to reliability, the MSForm is expected to produce similar results if it is distributed more than 2 or 3 times, or under different circumstances [130]. To support this claim, intraobserver reliability was detected during the interviews, as the respondents mentioned and sustained their survey answers. Moreover, by first asking the individuals what drives complexity and then observing their rankings of complexity indicators, alternate form reliability is also in place.

3.3 Interviews

As in the previous section, the reasons for conducting interviews and how data was collected and analysed are presented.

3.3.1 Goals of Interviews

To further clarify the answers collected via the survey and create a more comprehensive view of what an adequate or sufficient combination of complexity metrics that can accurately reflect the level of complexity is, as well as what types of costs can be connected to these metrics, two types of interviews were performed. The first one was connected to complexity drivers and metrics, whilst the second one was inclined towards the economic pillar of the organisation.

Interviews can be differentiated by the degree to which they are structured, thus having structured, open or semi-structured interviews. For this thesis, semi-structured interviews were used. These allow the respondents to bring new, unexpected topics to light [134], offering new angles for the development of the measurement method of complexity and understanding its cost implications. Hence, several open-ended questions and a topic list have guided the interview process.

The first type of interview, on complexity drivers and metrics, had the overarching goals of delving deeper into the ranking of complexity indicators identified with the help of the survey and of understanding what requirements stakeholders have for a measurement method of complexity (e.g., how they would apply a formula for quantification of complexity, from where would they like to gather data etc.). To reach these goals, the interviews started with an introduction of the interviewee and a description of their daily tasks, such that an initial impression of possible areas in which the formula can be used is created. Then, a discussion on application landscape complexity drivers and their measurement aims at uncovering which combination of metrics should be used to quantify complexity. Possible direction and requirements for creating a method to quantify complexity were investigated

during the interviews (e.g., easy to use, functionality, applicability). 16 such interviews were performed.

The second type of interview is aimed at discussing the costs that can be connected with the previously identified drivers of complexity. This helps with crafting an initial link between application landscape complexity and the costs emerging during change processes. As described in the literature, the "cost of complexity" is believed to be connected with the number of links and interactions of components, however, this should be explored. The desired outcomes are developing a cost breakdown and understanding existing touch points between ALC and different types of costs. 16 interviews on costs were conducted with people who responded to the survey.

During both types of interviews, if the direction and time allowed the exploration of the other topic (e.g., during a complexity interview, questions on costs of complexity could be asked and vice versa) the interviewees were encouraged to voice their thoughts. This way, a broader audience could give inputs on the two topics: the measurement of ALC and the cost of complexity.

3.3.2 Process for Conducting the Interviews

Data Collection

The questions for the interview on complexity metrics are included in Appendix D.2, whereas the ones on costs are part of Appendix D.3.

The interviews have been conducted either online or in person, adapting to the preference of the interviewee. During the in-person meeting, written notes have been made. For the online meetings, the interviewee was asked for consent to record the conversation.

Hence, to collect data from the semi-structured interviews, notes were taken during each interview, which later were enhanced by semi-verbatim transcripts generated with the help of the recordings [135]. The notes create a structure for the collected data, making it easily accessible.

Data Analysis

Once data was collected, open, inductive and thematic coding of the transcripts were performed. This all falls under the umbrella of content analysis, which *"involves coding data in a systematic way in order to discover patterns and develop well-grounded interpretations"* [136, p. 191].

Open coding started with the analysis of the first interview and the creation of a coding scheme which expanded throughout the interviewing process. As technology evolves, tools such as Atlas.ti can also assist with coding [137]. For this research, the notes taken during the interviews were also uploaded to Atlas.ti and by leveraging the artificial intelligence (AI) feature, the manually identified codes were compared and contrasted with the automatically generated ones. The data resulting from the analysis was grouped into themes with the help of inductive coding.

Paper [138] describes the step-by-step approach of a thematic analysis that can help set the basis of a conceptual model in qualitative research. Starting with reading the transcripts, keywords are highlighted and then codes are created. These codes fall under themes. Based on keywords, theme and code interpretation, models can be created. Image 3.2 illustrates

how one can arrive at a theme from several keywords grouped into codes. The results of the analysis process, grouped into themes, are presented in section 3.4.2.

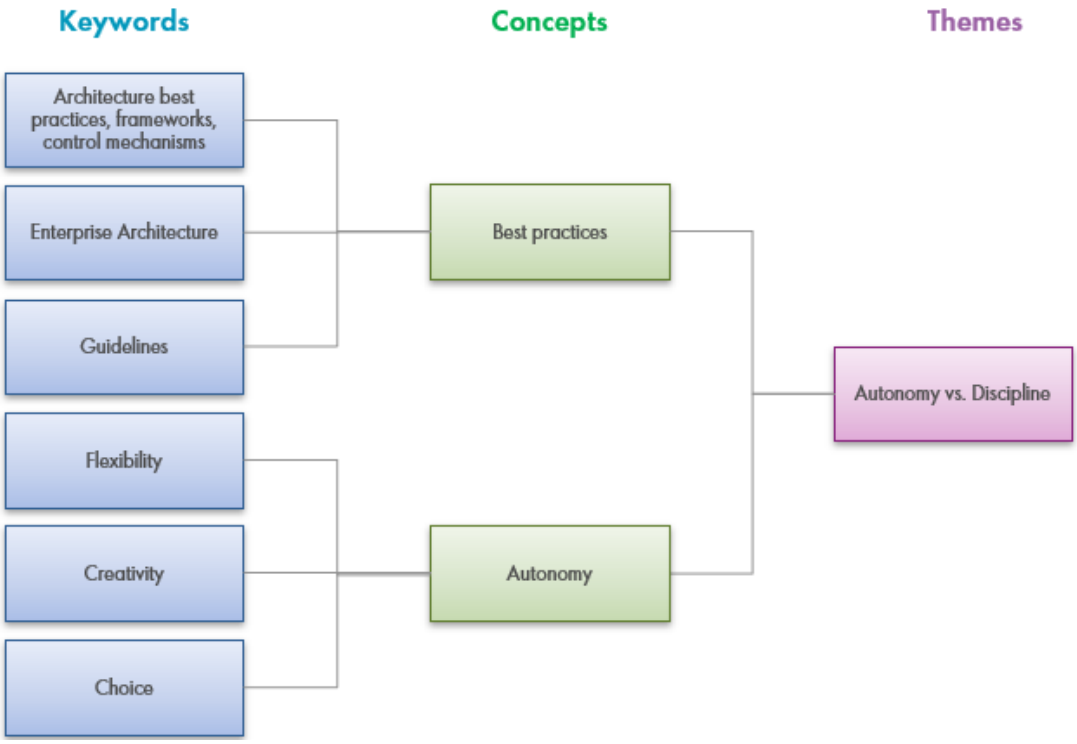


FIGURE 3.2: Process of Arriving at a Theme

Data Quality

Same as for the survey process, the validity and reliability of the interviews were considered. A debriefing procedure was in place, meaning that during the interview period, the experts part of the advisory board were asked questions to clear any ambiguities or identify biases in the outcomes. The coding process and the search for themes, patterns and variations can also help with the validity of the study.

3.4 Analysis and Results

Section 3.4 details the findings streaming from the answers of the survey and the interviews. Based on these, the section ends with an overview of some essential outcomes that guide the development of the artefact.

3.4.1 Survey

The survey yielded 32 responses out of a distribution pool of 38. In the IT area of the organisation, concerned with architectural complexity, the following roles were sought for insights: IT managers and different types of architects. The majority of the answers were received from lead architects (18 responses – 2 of which are lead data architects), then solution architects (7) and data architects (3). In terms of IT management, one response was received from a fixed IT manager, and another one from an employee with a dual role, serving as both an enterprise architect and an IT manager. Lastly, a strategy and planning, as well as a contractor provided their insights on the topic of ALC.

Introduction to Application Landscape Complexity

The application landscape of the enterprise at which the case study was conducted is perceived as complex by 31 out of 32 respondents. As in the survey, the question was a closed one, "Do you perceive the application landscape as complex?", the response, especially the reasoning behind it, was further discussed during the interview.

However, the participants were also asked about the elements that drive the complexity of this application landscape. To group the myriad of drivers for application landscape complexity, one respondent proposed a division between "technical" and "organisational" drivers. This division can guide the approach in which the answers to the survey are summarised. Furthermore, also in literature, as seen in Chapter 2, the causes of complexity can stream from both the business and the IT.

The survey respondent who described the application landscape as simple mentioned that this is due to the use of a hub-and-spoke model. This answer will further be clarified in the subsequent interviews, as it can mean that for a particular area of business, having a hub-and-spoke model is perceived as less complex. Even more, this can be connected with the view that point-to-point connections between applications represent the majority of the IT landscape of the organisation, as the other respondents did not mention using this architectural approach.

Technical Side

For technical aspects, the most mentioned element (25 out of 32 times) was the presence of too many applications. The respondents mentioned that the IT landscape is comprised of numerous applications. This value is considered to be influenced by the number of custom developments and flexibility, constraints in technology (e.g., use of legacy applications on local data centres) or usage of off-the-shelf applications. Next to these, there are redundant components or ones that cover the same capabilities, processes, and functionality. They can be "slightly different but similar" according to one respondent.

Next, the number of interfaces is important to mention. The respondents consider this an essential aspect that influences the ALC. Furthermore, a higher number of interfaces is commonly associated with point-to-point (P2P) connections which create a so-called spaghetti architecture [139], [140] in the company.

Data issues such as poor data quality, hard to combine various data, and the number of data stores were interestingly not only brought up by the data architects but also by the solution and lead architects. The fact that there are so many point-to-point connections makes it increasingly hard to exchange, update and maintain data. The data architects emphasized the insufficient coverage of data standards "for master, hierarchies and transactional data".

Some of the other discussed drivers are the large footprint of legacy applications, which can generate problems such as rebuilding applications or completely searching for new comparable products, the number of projects and products or digitisation.

Organisational Side

Moving to the organisational aspects, it was interesting to observe that these were mainly discussed as a cause for the growth (in size) of the application landscape. For example, having autonomy for a client or insufficient enforcement of data standards can lead to the use of more applications or the creation of customisations to adjust to a specific context.

The complex processes of the organisation which can entail business-specific solutions,

modifications to align with other companies or local legal requirements, variation and the coordination of these are influencing complexity. The complexity is emerging from having a large enterprise with "disconnected processes", as one of the respondents mentioned.

Even more, the governance structure/set-up of the company (e.g., business and operational models, requirements etc.) is leading to strict demands and budgets from the business. This complicates the enterprise-wide cross-business alignment. Two respondents indicated that the history of the organisation is important as this does not allow the "ideal" alignment between business and IT. This is extremely important, as, for example, when working with legacy applications these need to be integrated with new developments, thus requiring business process simplification alongside IT simplification. In literature, it is usually highlighted that organisations need to embrace an evolutionary approach to their landscape given that rewriting is never the key to success [140], [141].

Furthermore, having a company with an "over-engineering" background leads to the use of numerous applications and technical debt. For example, if architects do not want to work with legacy systems and they are creating changes on top of the current architecture, this becomes increasingly harder to change in the future. Even more, if certain people create solutions prioritising only what they know best (and making an abstraction of a holistic landscape understanding), the landscape will become unclear to the larger population and less agile in adapting promptly to future changes.

These two issues (governance and over-engineering) connect to change management, as a driver for complexity. The modification of the application landscape is a difficult endeavour.

Another interesting outcome was that legislation (discussed by 7 respondents) is considered to play an important role in complexity. The global and local asks and requirements can lead to the use of more applications or data structures, which can in turn generate data issues.

Some other organisational aspects discussed were: the lack of documentation, lack of skills of employees, or the process of mergers and acquisitions (M&A). According to Conway's law [142], the "technical structure reflects the social boundaries of the organisation that produced it", and based on the survey it appears that this law holds for this research.

Views on Complexity

By looking at the overall drivers of complexity and their impact on the application landscape, both positive outcomes and negative ones can be discussed. Having a higher degree of freedom and flexibility, for example, can lead to having more customised solutions in a landscape. With more customisations, some respondents mentioned the fact that better-suited solutions will be in place. These can generate value for the customers at the present moment. However, in the future by having these customisations it might become harder to adjust and understand how to integrate various applications.

One other example, are the rules and regulations for specific regions in the world. The organisation must respect them and be compliant even if this might mean an increase in the size of the landscape.

Hence, architects must make trade-offs when designing architectures and seeing the short and long-term benefits. This is supported by one of the participants who mentioned "having a maximum flux of complexity" that one needs to accept. Nonetheless, even with these trade-offs in mind, two respondents consider complexity to bring several benefits as it "promotes evolution ". By having complexity in the application landscape, it can be found

what works the best. Also "innovation stagnates" if a landscape is simple and a quicker turn-around time can not be reached according to a respondent.

When asked about the challenges of the ALC, all respondents agreed that higher costs for managing and operating the IT landscape are present. Furthermore, there will be less flexibility in adapting to future changes and difficulties in including innovations. Other factors were risk, vulnerabilities, data unclarity/mistrust/leverage, more time to get a person knowledgeable about the landscape (meaning more time and resources spent and consumed), ensuring interoperability with the ecosystem and time to market.

Figure 3.3 depicts the summary of the findings explained above.

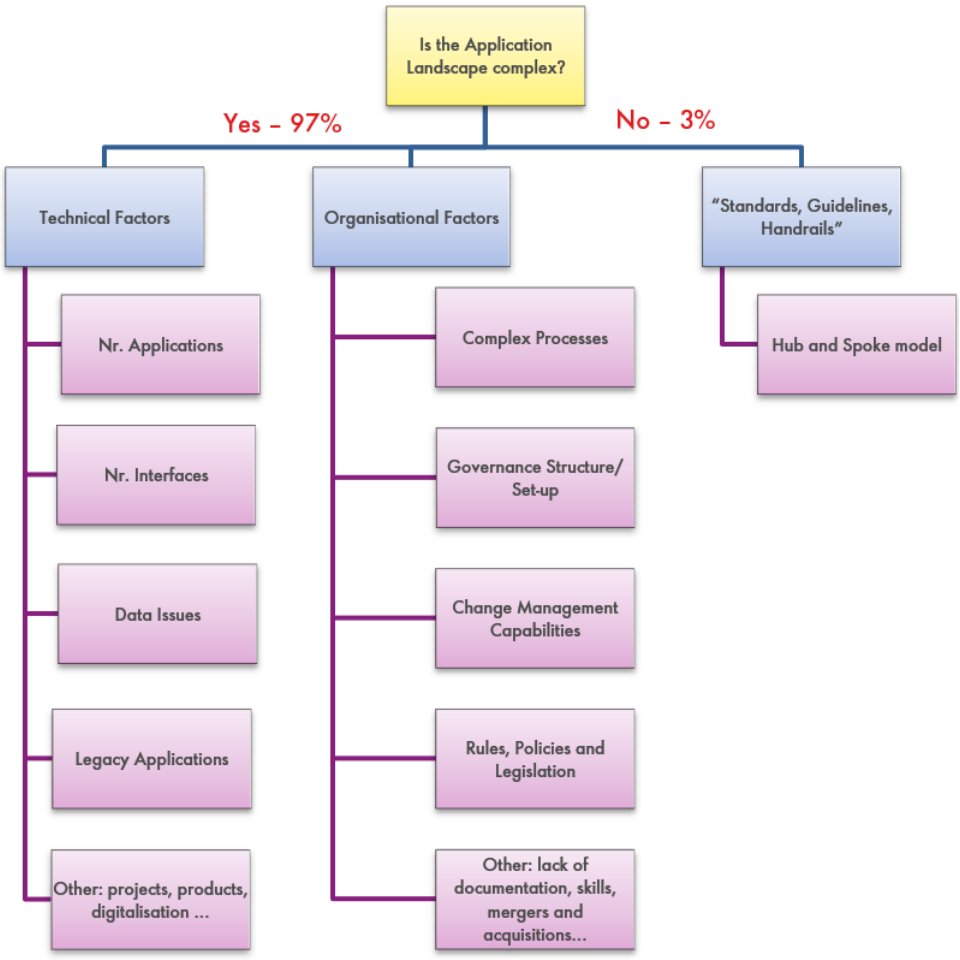


FIGURE 3.3: Survey Outcomes - Complexity Drivers

Complexity Management Capabilities

Upon understanding how application landscape complexity is viewed by the survey respondents, it was important to understand the current capabilities of the organisation when it comes to managing the complexity. More than 70% of respondents (78%) consider that control and management of complexity require improvements. A better understanding of why the remaining 22% of the respondents view complexity as an aspect which is sufficiently under control is expected to be highlighted during the interviews.

Definition of Application Landscape Complexity

The company uses numerous initiatives that discuss complexity or the need to reduce complexity, such as portfolio reduction or future architecture strategies. The most well-known is a strategy that describes what the organisation strives for in terms of application usage in the future, such that the presently used applications can be adjusted according to the desired state. Despite all of these initiatives, some others were mentioned, for example, rationalisation by business capability or IT-enabled simplification. It is interesting to observe the intricacy created by the sheer number of initiatives discussing complexity and proposing handrails.

According to 81% of respondents, no standard definition exists within the company for the complexity of the application landscape. However, one is needed because it can help create a standard language or a "common understanding" to describe complexity. Once such an aligned definition exists, it will become easier to clarify the reality of complexity, create guidelines/handrails and strive towards an easier monitoring and change process. One respondent mentioned that "it can be helpful if it [the definition of ALC] can be referenced to an industry standard" for benchmarking purposes. Once such a standard definition is in place, it is essential to test it in various parts of the organisation and check for exceptions to the standard. This definition is the first step towards creating a measurement for complexity, according to seven of the collected responses.

The 19% of the respondents who mentioned that a standard exists were asked about their claim during the subsequent interview.

Complexity Capability Maturity Model

When faced with the complexity capability maturity model, the answers were surprisingly dispersed. Figure 3.4 illustrates this. 8 people placed the company at level 0. Interestingly, two of the people part of this group initially responded that complexity is managed within the organisation. However, this shows that they also consider complexity as an issue. Level one reached one response. Again, 8 people placed the company in level 2. Whereas level 3 had 7 associated responses and level 4 only 4 votes. What must be discussed further is why there are people considering the organisation's capabilities for complexity management higher than level 4, with 1 vote for level 5 and one vote for level 6. One potential explanation for this division above level 2 was discussed during the mock survey when it was revealed that people might consider anything below that "unacceptable". Another point underlying this dispersed view might be the absence of a standard definition for application landscape complexity (as discussed above), which makes respondents think of complexity through different prisms.

How to Quantify Complexity?

To understand where data that can help quantify complexity drivers is found, the survey asked what information sources and tools can be used for this type of measurement. For both the information sources and the tools and methods, a grouping into people and technical/systems can be made. This is similar to the drivers of complexity. Currently, information resides in both people and systems. For creating the measurement, it is believed that project managers, lead architects, the teams and stakeholders can all be consulted for retrieving information on complexity. Furthermore, experts can be consulted to create a solid measurement. From a technical perspective, Bizzdesign [2] was the most mentioned

[2]Bizzdesign - Enterprise Architecture Software https://bizzdesign.com/

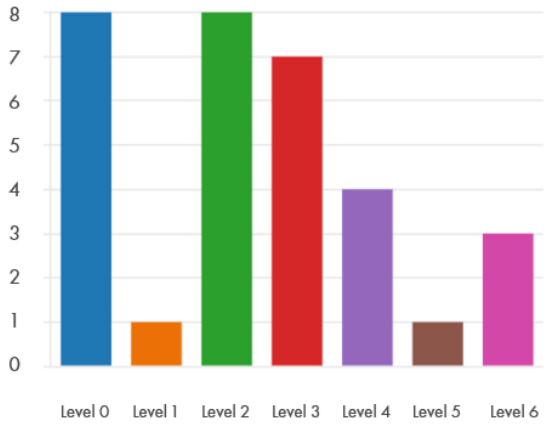


FIGURE 3.4: Maturity Model Results

platform, alongside ServiceNow [3], that can contain relevant information on ALC. In an ideal situation, in Bizzdesign, the designs of the application landscape are represented. This means that application components are depicted, alongside how they support the business capabilities, how they interact with each other (e.g., by using interfaces) and what technology underlies them. In ServiceNow, a list of all the applications in the company and their specifications is present. Thus, one can see the type of application (e.g., market standard, custom built etc.), the user count, the vendor or the business and IT owners, alongside the portfolio managers. Hence, by combining data from these two platforms judgments on indicators of complexity can be created.

Complexity Drivers and Costs

Going into more detail on complexity factors specifically focused on the application landscape, the respondents were asked to rank eight literature-identified indicators according to their preferences. The chart in Figure 3.5 shows the outcome of the ranking and it is in line with the previously self-indicated drivers of complexity: integrations, number of applications and customisation level. By looking at the first choice of respondents, the following order holds: number of applications (11), integrations (8), then customised applications (5). Interestingly, the lack of documentation was the first choice of 4 people, making it the fourth in this particular ranking view, however, it was in the last place for the overall category, with a 59%. The role of the people mentioning "lack of documentation" as an issue ranged from solution, lead architects and data architects. It is important to note that for leads or managers, this issue was lower in the ranking as compared to the rest. A discussion on the lack of documentation will be held during the interviews.

After ranking the factors, the respondents were asked for any missing elements. These, once again, fall under technical and organisational elements. For technical factors, the lack of a data model (and scattered data) that allows an easier connection between applications was brought up. Furthermore, the use of data and application programming interface (API) definitions to integrate applications or industry standards was mentioned. From an organisational standpoint, issues such as disconnected processes, ownership of applications, deployment strategies/tactical decisions, and the overall governance influence by data laws have been highlighted.

[3]ServiceNow - Cloud computing platform that assists companies in managing enterprise operations https://www.servicenow.com/

FIGURE 3.5: Application Complexity Factors Ranking

Moving to costs, in the answers to the survey, the commonly mentioned ones are part of the run and maintain category. These are mentioned by more than half of the respondents in various ways. Once again, a division between people/organisation and technical costs was observed. When talking about organisational costs, anything from the skill pool, manual work associated with development, design, testing, rework, and contractor costs, to the loss of clients and reputational damage costs were enumerated. One insightful response mentioned that costs associated with a change to a landscape fall under the following categories: decision costs, skill costs, analysis costs or learning costs. The technical costs detailed were: project costs, software licenses, hardware/infrastructure, duplication (e.g., paying for delivery of same capability), support, upgrade etc. Besides the people and the technical costs, risk and regulatory/legal concerns were said to influence the monetary value during change processes.

General Remarks

The last section of the interview brought to light that the majority considers a simpler IT connected to more agility for future changes. This will allow more modularity to support business requirements or a shorter time to market (value creation). Furthermore, a simpler IT is linked with a better user experience, a more efficient change management process and creating integrations in an easier manner (scalability, adaptability, maintenance). This is also believed to help with "lower cost of change". Nonetheless, some people said that "Complexity doesn't necessarily decrease agility if the team involved understands the complex landscape". One other answer was that skills are needed to manage complexity rather than get rid of it. The trade-off discussed previously, can thus be seen again.

For the thesis, 94% of the respondents consider a method for quantifying IT architecture useful. Hence, the artefact that is proposed by this research is valuable as it can help the organisation.

3.4.2 Interviews

Ways of Working

To further understand application landscape complexity, the IT or data architects (roles specific to the architecture department) have been asked about their tasks. This can make it evident if complexity, at an application level, is an issue or if it hinders any operations.

The IT architects, divided into lead or solution architects, are concerned with creating an

IT roadmap that can help achieve a strategic direction. This encompasses understanding existing opportunities in moving the IT landscape forward or collecting information about applications, data, integrations, business and technology to create enterprise architecture designs/models (e.g., ArchiMate representation in tools such as Bizzdesign). Based on this, an IT landscape "should be designed to accommodate and scale changes", as expressed by one of the interviewees.

Additionally, the lead architects are guiding and steering the solution architects. One of the respondents, part of the lead architect groups, mentioned that "Knowledge management or succession planning" is a task that must be performed. This connects with complexity as in companies, reorganisations result in shifts of employees. Hence if one employee leaves and their knowledge is not shared or documented, additional rework or sub-optimal solutions will emerge. These can potentially lead to the addition of new applications or the creation of unnecessary integrations.

Another interesting view of one of the respondents was that IT architects must balance opportunities and requirements, alongside dogma and pragmatism, as seen in Figure 3.6. This figure is created according to the respondent's view. Consider the following situation: A particular task or process presents itself and Product X has the capabilities to execute it. This represents an opportunity. The "dogma" within the company states that Product X should be the default choice (unless the occurrence of specific conditions). The architect can decide to do the task in this application. However, a more discerning approach should be adopted by architects. The requirements of a particular task need to be investigated, alongside the necessity of executing the process in Product X. This means critically evaluating the utilisation of Product X. As the respondent stated, "Understanding the two dimensions [in Figure 3.6] and the forces that pull us right, left and centre is exemplary of what I do as an architect". This means making decisions and accepting and understanding their consequences and impact.

The intersection between opportunity and pragmatism means an increase in complexity, as more customisations are expected to emerge, whereas the intersection between dogma and requirements can lead to technical debt due to short-term fixes. Interestingly, there could also be a link between customisations and technical debt on a holistic level, as with more customisations, technical debt could increase. Hence, the architect should be concerned with managing complexity for future agility and an easier change process.

For data architects, what matters most is understanding what data is needed for executing or realising business capabilities and "gluing it [the data] together". These types of architects are responsible for assuring that data quality is in place and that it can easily be shared between applications. From a complexity standpoint and a data lens, one of the respondents mentioned that "Complexity is how often we need to move data around to get the insights we want to have".

Autonomy versus Control

Building up upon the matrix seen in 3.6, a recurring theme was the level of autonomy or the discipline part of an enterprise. It is necessary to create the right level of autonomy, which gives architects flexibility in being creative, but also imposes certain standards, and guidelines that must be followed. A respondent mentioned that "complexity promotes evolution" in the sense that if an organisation encourage people to be creative and allows more solutions to emerge, one better system might be discovered. This system can be used for the whole IT landscape in the future. However, without certain handrails and controls,

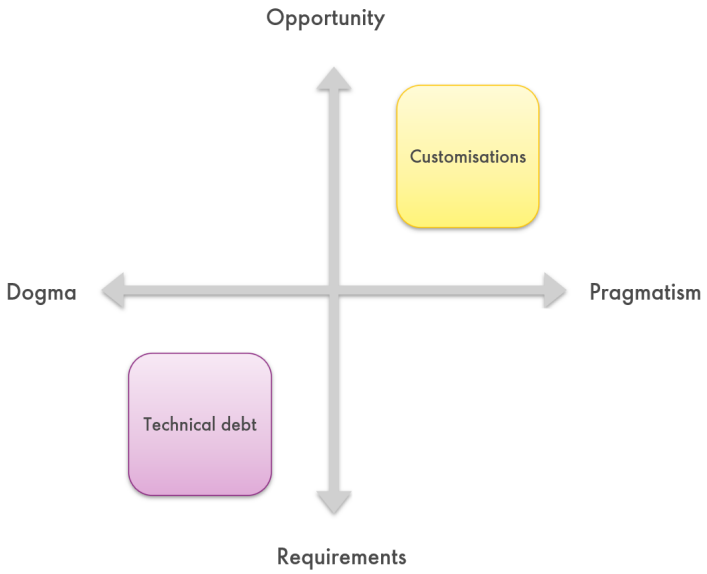


FIGURE 3.6: Balancing Matrix

this way of thinking can drastically impact the overall enterprise, as people can just make whatever choices they consider best.

To further underscore this factor, the simplicity of an IT landscape was commonly associated during the interviews with standards and guidelines. One example was given during one interview regarding shadow IT. This represents the systems being deployed to meet urgent needs and requirements, without a central governance acceptance. The respondent mentioned that when it comes to having such solutions, a clear direction must be in place to remove this and stop it from reoccurring. Furthermore, respondents agreed that architectural techniques selected and discussed at an enterprise level (e.g., using a hub-and-spoke model, using data products, using APIs etc.), if implemented by the majority, are making the IT landscape simpler and ready to meet future changes.

Levels of Observation

As observed above, complexity management is one of the most important aspects of the job of an architect. Enterprise architecture is a balancing act, considering trade-offs between factors such as autonomy or discipline. However, in big organisations, managing the complexity of an IT landscape, with an emphasis on the subset focused on applications, involves having a holistic understanding of the enterprise.

During the survey, given the complexity capability maturity model, as seen in Figure 1.4, it became apparent that only a few people are actively considering the importance of a holistic view of the enterprise. Upon seeing the diverse placement of the responses from the survey, as seen in Figure 3.4, the interviews clarified the reasons behind individual choices for each level. This dispersed view is created due to the absence of a clear definition for complexity drivers or the variety in the lines of business (LoBs) and the ways of working.

With the absence of a definition for application landscape complexity or for its drivers, subjectivity plays a role in how the respondents placed the company on the complexity maturity model. If an employee simply says that ALC means the number of applications in a landscape, then quantifying this number is possible (meaning level 3 in the maturity

model). However, if another respondent sees ALC as the number of interconnections, the number of applications, the variety of technology use or any other factors, these elements are not easily quantifiable. Hence, the placement in the maturity level will be lower.

Furthermore, the different operations that teams perform in an enterprise can influence the rankings in the maturity model. For example, in the field of human resources, more streamlined and market-standard applications are available, so complexity can be seen through the prism of disparity. This means analysing if the landscape is comprised of more or less market standard or customised applications. For LoBs that are working with specialised processes or should deliver customer-specific value, more customisations are expected to emerge. Therefore, in the first case, complexity is easier to manage, whereas in the second complexity is harder to evaluate.

From a people's perspective, the experience one has also plays an important role in the relationship with the complexity management capabilities. Some employees can be focused on their own area of work and consider complexity under control, whilst others can decide to look holistically at the organisation and see how their area's complexity fits into the bigger IT landscape.

The general outcome regarding the complexity capability maturity model was that level 0 was interpreted as the organisation not being aware of its complexity. Level 1 is the acknowledgement of complexity, level 2 means having initiatives to describe complexity at a qualitative level and from level 3 onwards people can start looking for indicators of complexity.

Complexity as Inherent Aspect of Business

The complexity of an IT landscape is "an inherent aspect of the business" as explained by one of the interviewees. This view was brought to light numerous times during the interviews. The application landscape complexity is backed up by organisational aspects, which are driving complexity. For example, the size or the diversity of elements part of an application landscape is present due to the company structure, culture or the external environment in which the company operates. One interviewee explained that "Technical aspects [application landscape elements] are indicators for complexity, but you need to go a level deeper to understand that the organisation impacts the technology".

Another illustrative example can be the emergence of duplication in application functionality or data overlaps, given that application owners could be "attached to their own applications", as another interviewee mentioned. This means, that if they move positions in the company, which is a common practice, they could try and stir towards the use of their preferred solution even if a similar alternative exists. The problem can appear due to their familiarity with the application over the years.

In the process of simplifying the IT landscape, according to another interviewee, "30% of the attention should be spent on IT and 70% is the business". In large organisations, with a history of projects, it can be the case that employees are selected for their "brilliance". Hence, over the years, implementations of IT were retrofitted or tailored in specific ways given the choices, preferences and ways of working of an employee.

At an application level, complexity emerges if the business asks are simple and IT starts to deliver something intricate, which inhibits the ability to change in the future in an agile manner.

Indicators of Complexity

To funnel in the application landscape and possible indicators of complexity, the interviews explored connections between such factors in terms of how they influence each other and how they influence the growth of complexity. The starting point for the discussion were the 8 indicators in the ranking produced by the survey, as seen in Figure 3.5.

The most discussed elements were: the amount of integrations or interfaces, application connections with business capabilities, followed by data issues, or customisation of applications. However, all the interviewees agreed that the indicators are closely "related", "go hand in hand" or that they can reinforce each other.

Applications

To start with applications connected with business capabilities, more than 50% of the interviewees mentioned that "more than one solution serving a target business capability" is an indicator of complexity. A distinction here needs to be made between business capabilities and application services. In enterprise architecture, to achieve a certain capability you can have various application components with completely different functionalities. However, if you have an application service performed by more than one application this is where complexity can be observed, as two components are performing (some of) the same services. In this case, the architects need to choose a tool that does the task "well enough", as described by one interviewee and move away from the "choosing is losing" mentality. The implications of the change must be acknowledged and an informed decision can be taken.

Interfaces

Next, in a complex IT landscape, the amount of applications is not an accurate indicator, rather the amount of integrations between applications must be considered. The growth in the number of applications, in the current setting of the case study, generates more functionality overlaps. Given that, the number of interfaces is expected to increase. Having point-to-point connections, was also a concern brought up by the interviewees. This creates the so-called "spaghetti architecture" and increases the overall complexity of the landscape. The use of generalizable interfaces can help with the decrease in the number of point-to-point connections, as stated by one of the interviewees. The reuse of integrations is extremely important and can affect data overlaps, as mentioned by a solution architect "The integrations have a direct effect on the duplication of data". One of the data architects mentioned "The unclarity of the integration, what is used, what is already integrated and what can be reused needs to be addressed".

Data

When talking about data issues, complexity primarily comes from different components using the same data entities. The data must be shared between applications in a seamless manner, however replicating it in multiple systems creates intricacies of modifying or combining the data. Having coherence in the data structures decreases the overall application landscape complexity. Solutions for having a better data structure were said to be the use of data meshes (as two of the respondents stated out) or the use of data platforms and products. In these architectural structuring views, the common underpinning is having components that can be reused. In the enterprise, it became apparent that data ownership is an indicator of complexity, considering that the same data can be found and used in different systems. Data mesh addresses this concern by considering the nature of ingested data and making the data the primary organisational concern [44].

Customisations

For customisations, ideas such as "architecture stepouts from standard development patterns", "building new applications in-house and replicating functionality or creating overlaps in data" or "changing the configurations of a commercial application" have been discussed. When working with vendors, if a tweak in functionality is made, a respondent mentioned that "It would be easier to change your processes to fit the application, rather than have the inability to change in the future due to a lock-in". Furthermore, if one employee has changed the constructing elements, generating a higher variety, the knowledge will reside within the person making the change in the majority of cases. The changes made in-house/customisations are believed to increase complexity, as sometimes integration problems emerge or functionality can be easily replicated. If an organisation wants to differentiate and create customer value, a trade-off analysis needs to be performed.

Lack of Documnetation

An interesting outcome was that the absence of documentation, or the age of the documentation was not considered a direct indicator of complexity, but rather a factor that impacts "everything" as described by an interviewee. In case the documentation is inadequate, architects might not know what integrations can be reused, leading to a higher number of integrations, or spending a lot of time in getting the "truth about an architecture". The lack of documentation, from the eyes of a contractor, is "not really about complexity, but it is about hiding how complex something actually is, making it more difficult to understand".

The relationships and links presented here are discussed in more detail in Chapter 4.

Quantify Complexity

What would be an appropriate way and means to quantify complexity using the above-mentioned indicators was a topic of discussion during the interviews. As the scope of the thesis is scaled down to projects and quantifying ALC, the desire of interviewees to create a tool that can look at an organisation holistically will not be considered. This means, having an overview of how a change in the project impacts the overall IT landscape or the portfolio is out of scope.

To quantify complexity, employees desired to apply a model-based approach, thus, drawing insights based on enterprise architecture designs realised by architects.

One of the first requirements is to start with a simple tool for quantifying complexity, adhering to the "Keep it Simple Stupid" idea, as mentioned by a respondent. Complexity must be defined and broken down into indicators that are well-described. If a tool is simple enough, through pilot-tests it can be adjusted and refined. This way, in the future the tool can be used even for benchmarks (as envisioned by managers).

Once the elements that encompass the application landscape complexity are defined, the next important aspect is to mention what information needs to be in place for a correct model-based analysis. For example, having sufficient information documented about the type of interface.

The tool should help with effective decisions, hence, it is requested to quantify complexity and translate the value into more than just numbers, "the formula must give a feeling of direction" or "convince people to select a solution", as per the ideas of the lead architects.

Additionally, the respondents mentioned the desire for minimal user input and the use of

as much automation as possible for calculating complexity. The results should be displayed using dashboards, charts or any visual representations. These requirements could be addressed if a tool such as Bizzdesign would contain all the information needed to quantify ALC. Presently, more discipline and information are needed to effectively use Bizzdesign as a solution option tool and quantify complexity. Nonetheless, the participants in the interviews were enthusiastic about its potential. By leveraging Bizzdesign, the indicators of complexity can be quantified per solution, and by doing so, architects can revisit their choices and easily compare future solutions. In Chapter 5, a proposal on how Bizzdesign could be enhanced to assist in solutions options for effective decisions is introduced.

Costs

At the moment, in the organisation at which the case study was conducted, a large emphasis is put on costs. There are budget cuts or desires to rationalise portfolios. Hence, more than 50% of architects consider that simplifying the IT landscape, from an operational perspective is going to reduce costs. For example, from three applications currently part of the landscape which might perform similar or the same tasks, they consider that only one can be chosen, therefore reducing the total cost of ownership. The visible costs and the ones taken into consideration when changing are normally the project costs or the run and maintain costs.

However, when performing a solution options analysis, besides the investment or the run and maintain costs, there are also several "hidden costs" which could appear driven by a complex application landscape.

As architects, getting a grasp on the costs of a landscape is an intricate task, as many people from within the organisation need to be contacted to create a clear understanding. Several types of costs that can be accounted for were mentioned. Figure 3.7 is an updated version of Figure 2.3 which highlights in red the costs discussed during the interviews (some additions as compared to the literature). For the interface costs, one can also think of costs due to integrations and APIs, as a respondent mentioned "the number of integrations multiplied by their costs divided by the number of applications must be lower than the total investment in a project". When talking about licenses, these can be connected to the overlap in functionality, if one pays for the same service twice.

During the interviews, it was clear that the invisible costs were the ones concerned with people and the opportunity cost. The rest of the elements in the figure can be estimated or approximated at the start of a project. During a change process, the costs generated by the complexity of the solution are not considered. The contractor who was interviewed said that from his experience, often "people look at how much something costs now, not how a choice will affect costs in the future". For example, if somebody makes a change now, how much effort will it be required to adjust errors that may occur, are there sufficient resources available to fix it (both in terms of people and technical resources)? Will there be people to work with the change? Should one take the path of least resistance such that a solution will be easier to maintain? This was also showcased by a lead architect saying that "typically... we just look at license costs and implementation costs, we don't take into account all the other costs that are necessary like discussions, meetings, getting business people to put the time in the solution." The "risk appetite" of an enterprise, alongside the business requirements and necessity should also be factors for the opportunity costs. Lastly, the possible costs of rework need to be visible, as people consider that spending more time upfront will decrease the time and resources spent in the future.

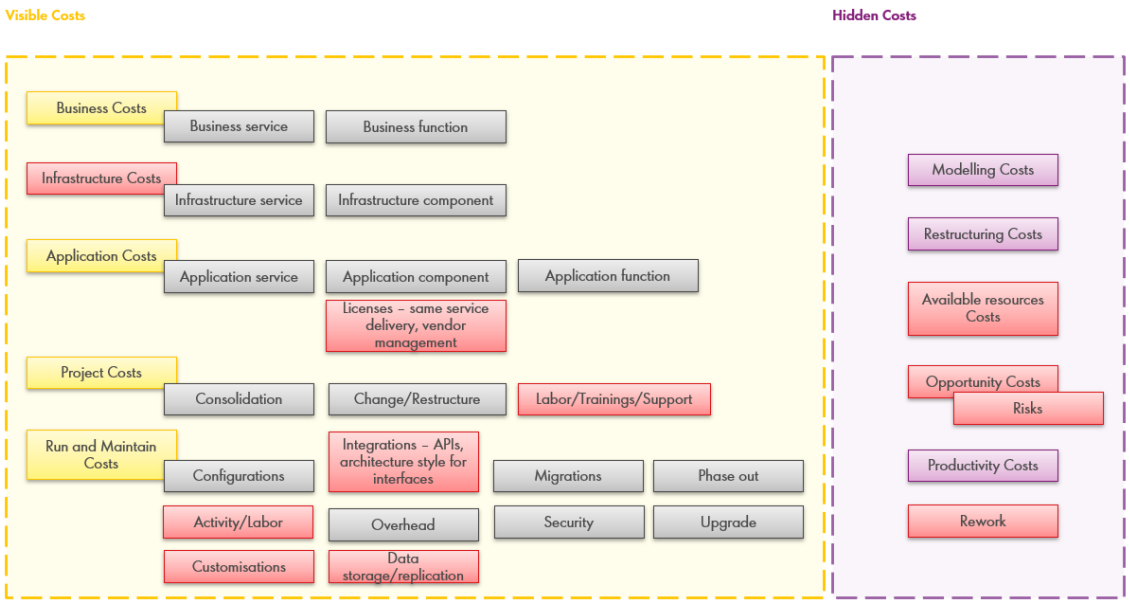


FIGURE 3.7: Updated IT Landscape Costs

The drivers of complexity and the indicators can have a relationship with an increase in costs. The most straightforward link is with the run and maintain cost, as previously discussed. Three interviewees tried to connect three pillars: time, cost and complexity, by saying that if something is quicker to be built (less time spent on making a change, maybe due to imposed timelines), this will result in more costs for maintaining and more complexity. This approach is similar to the idea behind the project management triangle [143]. Figure 3.8, looks at the project management triangle from a complexity lens.

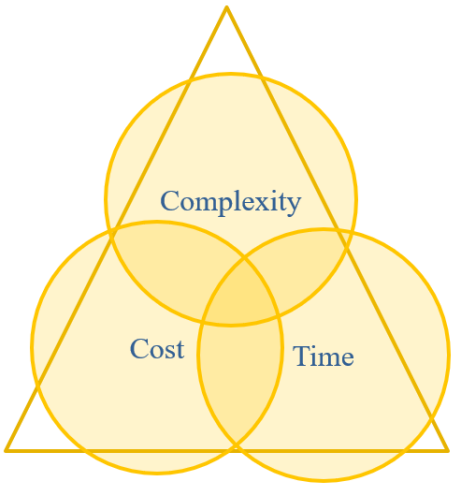


FIGURE 3.8: Complexity Management Triangle

3.4.3 Overview of Findings

The first step towards a simpler IT, meaning more agility to future changes, allowing an enterprise to do tasks that they could not normally do, can be taken by "easily fixing the

IT environment" (as mentioned by a solution architect) and looking at the applications functionality overlap, the data overlaps and the number of interfaces. Later, to maintain this simpler IT, the enterprise should delve deeper into the internal and external business drivers for complexity.

To showcase the essence of the survey and the interviews, an issue tree has been designed, trying to closely align with the mutually exclusive, collectively exhaustive (MECE) principles [144]. The elements appear in singular categories and when connected they form a holistic understanding of drivers and indicators of ALC. The indicators of complexity are the ones on the lower tree levels, whereas the drivers are the ones closer to the root of the tree.

Given that the insights are collected from a sample of 32 individuals, it is expected that other branches can be added to create an all-encompassing view on application landscape complexity.

Figure 3.9 illustrates what drives the complexity of an IT landscape, based on the responses collected from the interviews, survey and the literature. The high-level drivers (depicted in the green boxes) are also added as a reference to indicate by which means they have been identified.

To model drivers and indicators of complexity in ArchiMate, the motivation layer can be used, as seen in Figure 3.10. In ArchiMate, according to the latest specifications (3.2) [1], the following definitions describe the meanings behind what can be seen in the image. A driver (first element on the left in the figure), represents an external or internal condition that motivates a company to implement changes in order to achieve goals. An assessment (the elements in the middle of the figure, assigned a magnifying glass icon), represents the "result of an analysis of the state of affairs of the enterprise with respect to some driver" [1]. Lastly, a metric is a specialized driver, representing something that can be measured and linked with objects or relationships [145].

Drivers and Metrics of Complexity

The model resembles aspects from the Technology-Organisation-Environment (TOE) framework [146]. The TOE framework explains how the process of adopting and implementing innovations is influenced by three contexts: the technology itself, the organisational context and the external environment. Similarly, the model in Figure 3.9, encompassing the answers from the survey and interviews, portrays the drivers and indicators of complexity that can have an impact on the process of managing the IT landscape complexity of an enterprise. These are also grouped into three categories: the external business and internal business environment (similar to the environment and the organisation in TOE) alongside the IT environment (or the technology context from TOE).

The internal and external business environments are the ones that impact the size of the IT landscape, the presence of more interconnections, the diversity or the ability of an organisation to efficiently change.

External Environment

Starting with the external business environment, if the market is not mature enough, for example in case of new solutions for sustainability reporting (as described by one of the respondents), this can lead an organisation to develop in-house systems, which in turn can result to a higher percentage of customisations, problems with integrations in the future

or technical debt. In the case of new technologies being developed, problems arise when an organisation aims to introduce them. Adjustments to the current stack of technologies are needed causing a higher diversity of components or overlap in data and functionality.

For the industry characteristics, the diversity of clients and the customer-driven mentality of an organisation can create a diverse stack of solutions and a higher percentage of customisations to meet the client's needs. Furthermore, if a vendor lock-in emerges, the enterprise may be faced with technical debt, higher costs of rework, and more interconnections to integrate the solution used within the landscape.

Next, with laws and regulations, the IT landscape increases in size, as with a higher geographical distribution different compliance measures need to be considered. This also impacts the diversity of the existing applications.

Internal Environment

Turning onto the organisation itself, or the internal business environment in the model, the company structure and size, the culture, politics and people are elements driving the complexity of the IT landscape. In large enterprises, which operate in numerous countries and work with a variety of clients, LoBs are helping with coherent value delivery. However, each of the LoBs can use their preferred applications and without proper handrails or a holistic overview, this causes a lot of redundancy in the overall landscape. Moreover, as each LoB can have very intricate processes or strategies, customisations can be in place. Given the governance or the ownership of certain applications, legacy applications, which are preferred by owners, can also stay in place for a long period.

Mergers and acquisitions are also a common activity which is performed in large organisations. During such a process, the enterprise needs to decide between allowing the company acquired to keep their systems or onboarding the systems into the current stack. On one hand, maintaining the systems increases the size of the IT landscape, the overlaps in functionality and data and can cause heterogeneity issues. On the other hand, entirely removing the systems of the company acquired comes with challenges in terms of creating temporary solutions, having integration issues etc.

When talking about the company culture, if freedom is given to employees (as they are hired for their "brilliance") the expected outcome is that a more diverse technology stack will be used. This will be harder to change in the future. With the tendency to overengineer, the size of the IT landscape can increase, as well as the amount of customisations and integrations present in an organisation. This freedom can also lead to benefits as a solution developed in a side of an organisation can become a standard for others, and potentially decrease the size of the IT landscape.

When talking about discipline, a mentioned problem was the lack of documentation. Without adequate documentation, which can promote an easier understanding for employees, complexity is expected to increase. If documentation coverage is decreasing, the overall quality of documentation is low or the documentation updates are missing, complexity becomes hard to manage and understand.

Another interesting aspect to consider is the risk appetite of an enterprise. If people are afraid to make a choice and change, this only increases the size of the application landscape, alongside the number of possible customisations added on top of existing solutions. In such cases, when faced with transitioning towards a desired state, the effort will be significantly higher.

For company politics, if the enterprise imposes certain time scales, for example, business requires a quick solution, this leads to technical debt, as well as more variety in the landscape. If company reorganisations are encountered, a higher number of elements in the IT landscape can become visible, or more overlaps, as people tend to move with the knowledge that they acquired.

Lastly, the people and the employees, have a great impact on the IT landscape complexity. If people are not following certain guidelines, they are unfamiliar with the existing landscape, or have a mindset in which agility is not considered, the size, diversity, an interconnections of an IT landscape will continuously grow and become difficult to manage.

IT environment

For the IT environment, the size, interconnections, diversity and change rate are the main drivers for complexity, as described in the literature and identified during the case study. They can be broken down into several elements in order to arrive at indicators of complexity for the application landscape. These indicators are visible in the grey area and they are the input for the artefact development.

When describing the applications, different dimensions can be applied to the word. For example, an application can be seen as a product, as a piece of code or even as an operation. The work of Wang et al. [147], introduces an ontology for software. For this research, the current model refers to application as a product (which can be licensed or not), constituted by a system or program, and underlying code.

General Remarks

From a business standpoint, it can be observed that complexity can be perceived as both a positive and negative element. With a higher complexity, an organisation can attend to various clients' needs and maybe an indication of a high geographical spread can be given. However, it is also the case that a higher complexity is associated with a larger effort to integrate new solutions or create an effective change to the IT landscape. Hence, the right level of complexity needs to be achieved.

Going forward, for the treatment design, only a small subsection of the IT landscape complexity is selected for further investigation. This is concerned with the application landscape complexity, thus the elements from the IT landscape which can drive complexity. The yellow elements in the model, the indicators of complexity at an application level, are considered as aspects that can be tackled first by an organisation. These will serve as the basis for an objective view on complexity, based on which the "relativity lens" or determining if complexity is good or bad can then be created.

If actions are done in a timely manner, the level of ALC can decrease, allowing a more agile and simpler IT landscape. For example, when describing unnecessary variability, it is believed that both business and IT can identify which applications perform (some of) the same services and rationalise them. However, it is critical that drivers closer to the root of the tree are addressed and fixed, such that in the future, complexity at an IT level can be prevented.

3.5 Summary and Takeaways

Chapter 3 investigates the problem of application landscape complexity in the context of a large enterprise. Through a survey and subsequent interviews, the participants in the case study could detail their views on drivers and metrics for application landscape complexity, and its cost implications. Furthermore, the researcher could understand the current ways of working of the participants, such that the artefact design could be tailored to the present tasks performed in the company.

The core messages from this chapter are listed below.

- The complexity of an application landscape can be driven by both technical and business aspects. To funnel in, these two categories can be broken down into external and internal environments and the IT itself (similar to the TOE framework [146]).
- The ALC is an inherent aspect of the business. To showcase this, one can look at Conway's law where "technical structure reflects the social boundaries of the organisation that produced it" [142].
- Improving the ALC is something which will bring short-term benefits, but to avoid having a complex IT landscape one needs to solve the issues streaming from the business side.
- The role of an architect is to manage complexity for future agility (or other elements such as stability, security/risks etc.) and an easier change process. Complexity must be balanced, as it can be seen as both a positive or a negative aspect.
- Indicators of complexity such as the number of interfaces, percentage of customisations, functionality or data overlaps are believed to "go hand in hand" or to be "closely connected".
- To quantify complexity it is important to have a clear definition of what needs to be measured and have EA models that include the details on the indicators for measurement. For each company, this complexity needs to be defined such that the employees will have an aligned view on what is meant by it.
- Application landscape complexity is believed to create higher costs for managing and operating the IT and decrease the level of agility of an organisation when adapting to changes or when including innovations.
- When discussing cost and complexity, the people first usually consider run, maintain, and operate costs. The emphasis is rarely put on the prolonged decision process costs, opportunity costs or the costs of available resources in the future.

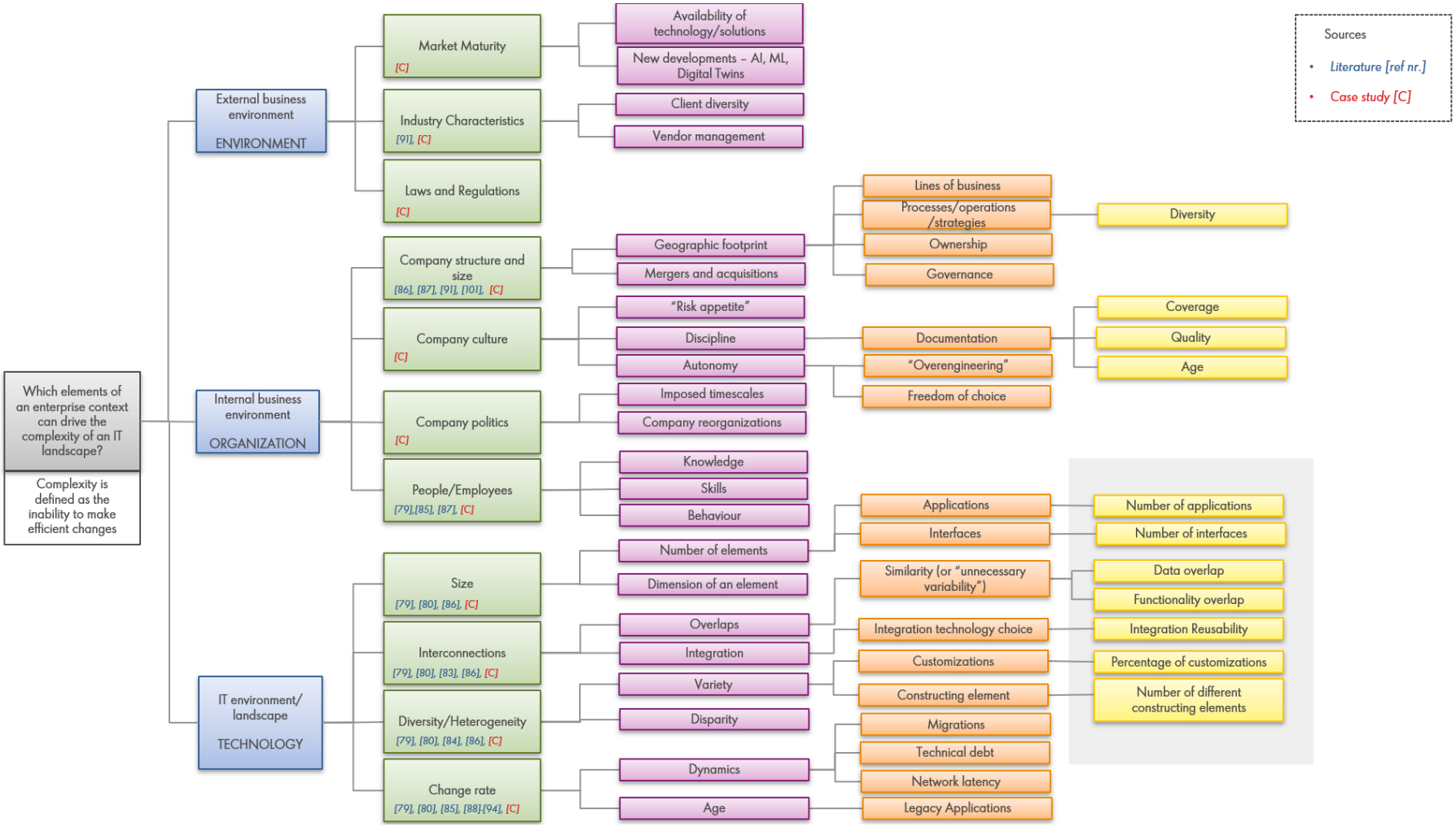


FIGURE 3.9: Application Landscape Complexity Overview

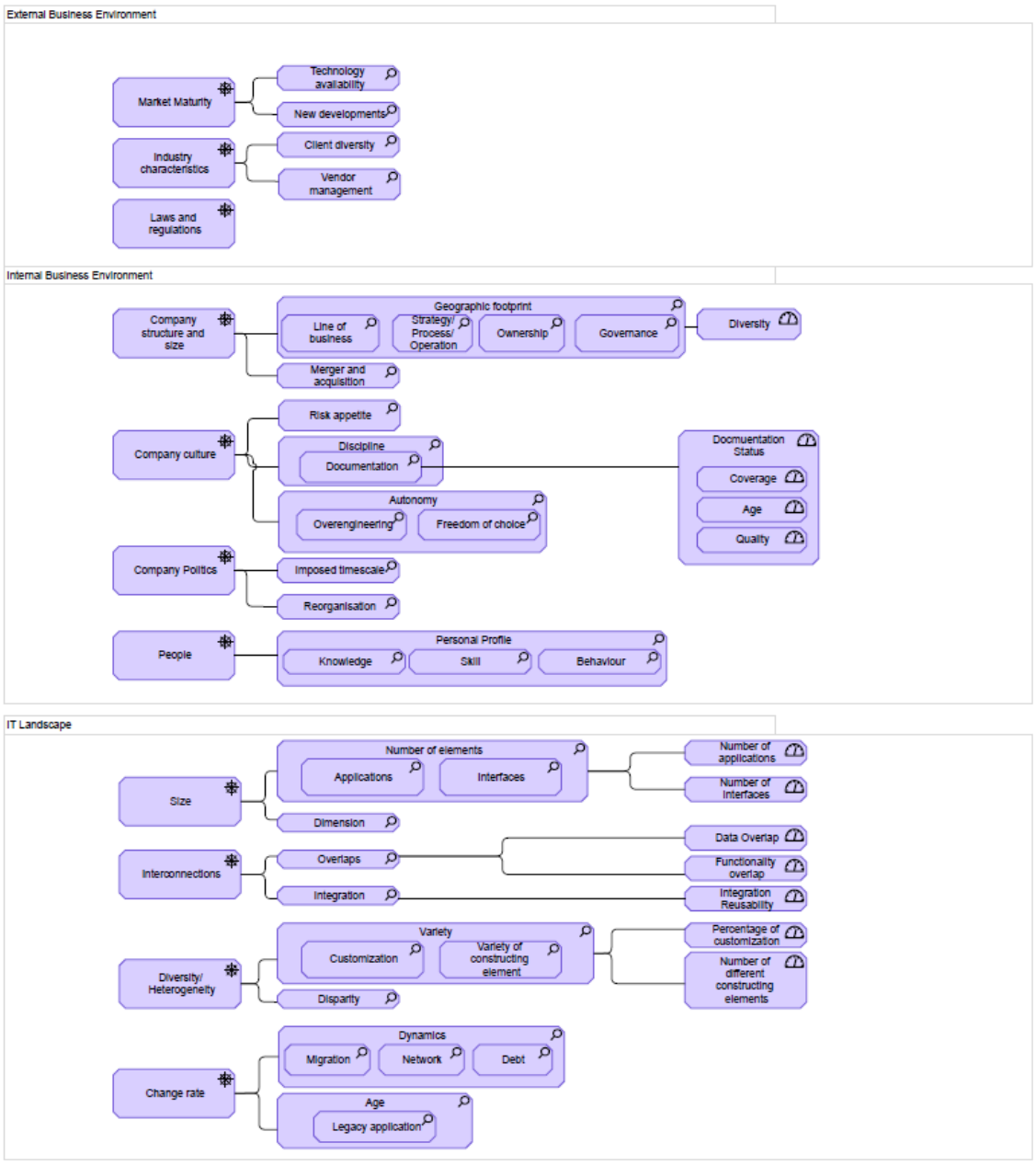


FIGURE 3.10: ArchiMate Drivers and Metrics for Complexity

Chapter 4

Treatment Design

The main goal of the research is the development of a measurement method for quantifying the application landscape complexity of large organisations. This should be done by analysing enterprise architecture designs. Hence, this chapter describes the design of the treatment for achieving the goal. First, in Section 4.1, the methodologies guiding the artefact development are introduced. Then, the overarching description of the artefact's intended use is introduced in Section 4.2. This is followed by the description of a measurement model, in Section 4.3, based on information gathered from the literature and from the case study.

The requirements of stakeholders for a method to quantify complexity are detailed in section 4.4 alongside the design choices for the final artefact and a detailed procedure on how the method can be used. The last sections 4.5 and 4.6 introduce the answers to the third and fourth research questions, and respectively an overall view of the chapter.

4.1 Measurement Methodology

To design a treatment, in the form of a measurement method, that can help achieve the goal of the research, a combination of three well-established works has been used. These are done by Abran [63], Petri et al. [148] and Gericke et al. [149]. A measurement is *"a process based on empirical interaction with an object, aimed at producing information on a property of an object in the form of a value"* [150, p. 24]. The measurement is designed on purpose, to support in achieving a certain goal or to make a property of an object visible.

4.1.1 Measurement Methods Development

Abran mentions that three activities need to be performed for the design of (software) measurement methods [63]. The first is defining the measurement principle, the second is concerned with defining the measurement method and post-design, the measurement procedures need to be explained (this last step is out of scope for the thesis).

The measurement principle is the phenomenon serving as a basis for measurement, in the case of this research, this can be represented by a measurement model in which indicators of complexity and their relationships are introduced. In this model, the indicators need to be clearly defined and unambiguous [63]. The entity under consideration, application landscape complexity, can be measured via the quantification of certain indicators, and an adequate model to characterise that is proposed in section 4.3.

The next step is the measurement method, which is a generic description of a logical sequence of operations used in a measurement. This step is linked with the proposal of a practical method, that could assist in quantifying ALC. By the term "practical", this thesis refers to the fact that the measurement method must be easy to use and apply in the context of an enterprise, thus in a practical setting. A discussion on how indicators can be measured, what falls out of scope and how to perform the measurement needs to be considered. Section 4.4 details the logical set of operations for measuring complexity.

4.1.2 Methodology Overview

An all-encompassing structured methodology for measurement development was created by Petri et al. [148]. Three main stages are proposed for the design of measurement methods: the planning stage, the execution stage and the interpretation stage. In the planning stage, the purpose of the method, the object under measurement and indicators, the measurement principle and the method are described. In the first stage, similarities with Abran's work are visible, as measurement principles, methods and procedures need to be designed. The next stage is concerned with executing the method, whereas the last one is connected with interpreting the results.

Gerike et al. [149] also explain that the description of a model should cover its core idea, the representation in which the information is described, the procedure to be followed and the intended use or selected tools. The representation and the core idea can be considered similar to the measurement principle. The procedure and the tool are comparable to the measurement method. The intended use details elements such as the purpose and the scope of the method.

Figure 4.1 shows the steps performed for designing the artefact in this research, as a combination of the ideas presented above.

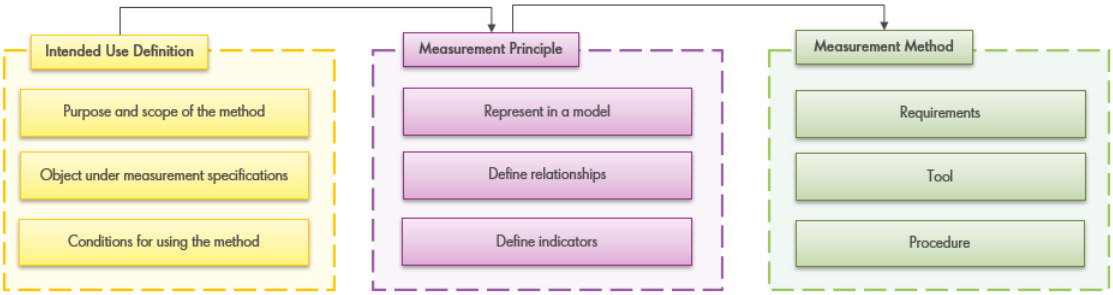


FIGURE 4.1: Measurement Methodology Steps

4.2 Artefact Intended Use

4.2.1 Arterfact's Purpose and Scope

The purpose and scope of the treatment to be designed is the measurement of application landscape complexity in enterprise architecture models. The measurement method should be used for assessing solution options for projects, and based on the assessment of complexity, a more informed decision can be taken by IT architects.

4.2.2 Object Under Measurement

Solution options in projects are setting out the situation, complication and needs of the business, alongside an analysis with several considerations for possible transformations. IT performs this solution options analysis and then informs stakeholders what the best way forward is. Thus, by looking at project solution options, evidence is provided regarding a choice for a change in the IT landscape. As application landscape complexity is not an indicator normally used for making choices, by having a value for quantifying complexity, a more informed decision can be taken.

Furthermore, by using several indicators to generate a complexity application landscape score, the architects can observe if there are structures, rules or patterns in a solution design which are associated with less complexity. These can be investigated to see if they allow improved agility at an application level (e.g., usage of reusable interfaces instead of point-to-point connections between applications).

Hence, the object under measurement is represented by enterprise architecture application landscape designs (e.g., ArchiMate models represented in tools such as Bizzdesign or drawings/designs of the application landscape in PowerPoint slides). The general property, or attribute, of these EA designs that need to be measured is complexity. This can be defined, for the scope of the thesis, as the inability to change or to efficiently adapt to a rapidly evolving environment.

4.3 Measurement Principle

To continue with the measurement context, for measuring complexity, which needs to be applied to EA models, the indicators which can illustrate complexity at an application level have to be detailed.

To do this, the measurement model in Figure 4.2 was created. It shows the relationships between the indicators of complexity at an application landscape level. The extensive literature reviews performed for this research and the case study insights helped with its creation.

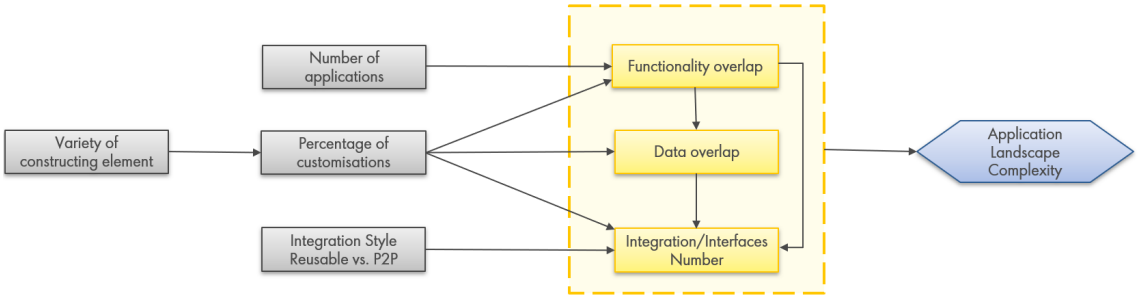


FIGURE 4.2: Structural Model - Application Landscape Complexity

4.3.1 High-level Model Description

Variety of Constructing Element

The first relationship in the model is between the variety of constructing elements and the percentage of customisations. If the number of applications which are part of the IT landscape are constructed differently, more customisations can happen during change processes. For example, if applications are using a myriad of programming languages, suboptimal implementations or customisations will emerge to ensure seamless communications between existing applications. With a more standardised foundation of elements, it is believed that the change process will be more agile.

The connection flows from the variety of constructing elements to a higher percentage of customisation. The customisation levels are not necessarily an indicator of having different constructing elements in a landscape. For example, several applications can be customised but this does not mean that the tweaks are done by using different constructing elements.

The variety of constructing elements has no direct link with application landscape complexity given that it is only driving some of the possible indicators of complexity when making a change. With a high variety, one can end up with a more intricate process of managing integrations.

Customised Solutions

Moving to the customised solutions, when trying to integrate systems built in different languages or having small differences in functionality, the number of integrations increases. Furthermore, a higher percentage of customisations can be an indicator for replication, as certain applications can perform similar or just slightly different tasks.

Customisation can also be created as a means to streamline processes in the case of functionality overlaps. For example, if two systems perform similar tasks, the organisation can make a choice and use the services offered by one of them. In this case, it can mean that customisations will be created to modify the existing processes to allow the selected application to take over the data in the other one.

With a landscape in which customisations are present, more data overlaps can emerge, as the systems can duplicate data which is present in others. This makes data hard to manage, update and maintain consistent. The reason behind this is that new data structures or entities will not be aligned with the current landscape.

Once again, for complexity, as the percentage of customisations is a problem introducing issues with variations in functionality, data structures or integration points, the decision was made not to create a direct link between the two (percentage of customisations and complexity).

Number of Applications

The number of applications drives the functionality overlap in an IT landscape. For example, if the number increases during a merger and acquisition (M&A), it is more likely that there will be applications with redundant features. Additionally, if different lines of business do not have a holistic understanding of the architecture of the organisation, choices of applications happening in one area which are not coordinated can lead to functionality overlaps.

Interestingly, the number of applications is not a direct indicator of complexity. Sometimes a landscape can be comprised of numerous needed applications, but the problem of complexity only arises if there are functionality overlaps, data overlaps or unnecessary integrations present. To support this point, during the case study, an architect mentioned the fact that the ratio between the number of applications and the number of interfaces would be an adequate measure of relative complexity. As this claim goes beyond simply considering the number of applications, it becomes apparent that this factor should be connected with the rest to indicate the complexity of the landscape (hence supporting the measurement of relative complexity).

Integration Style

The next relationship that needs to be described is between the integration style choice and the number of integrations. Integration style, in this case study, means observing the type of interface between two applications (one application providing certain data for the other to consume).

Two disjoint categories of integration styles guide this research, called point-to-point connections (P2P) and reusable integrations. These categories emerged after the case study discussions with the respondents. The first category means having only two applications using an interface (direct, tightly coupled connection between two applications), whereas, in the case of reusability, this interface can be used by multiple applications (loosely coupled, supporting multiple applications).

The first, P2P, reflects the use of an interface specifically designed to facilitate interaction only between two particular applications. These represent tightly coupled applications, depending on each other and making assumptions about how the others work [141]. When making changes in the application landscape, due to their close connection, when modifying one application the coupled application is affected. If one application called B, uses an endpoint of another application, A, this would be a P2P connection. This endpoint should only be used by B. For example, it can be seen B calls A, and A transmits data tailored to the needs of B. All the changes which emerge in Application A will be reflected in B. This P2P connection is similar to the use of "a purely internal" API [151] which can be used just for a specific purpose.

The second category, reusable interfaces, means that one application can have an interface consumed by multiple applications. This creates a loosely coupled IT architecture, which promotes an easier change process of applications and is similar to the use of public APIs [151], [152]. By considering the example introduced before, with Application B using an endpoint of Application A, if another Application C starts using the same endpoint, this would mean that the integration becomes reusable.

The work of Hohpe and Woolf [141] on enterprise integration patterns can also be used to compare and contrast the two disjoint integration styles. For example, the authors mention middleware, as common standard communication mechanisms between applications [141]. This can assist in making the integrations in a company more reusable (e.g., consider a hub-and-spoke architecture [4]). With different middleware patterns, promoting a more loosely coupled architecture is observed.

Furthermore, four application integration options are discussed by Hophpe and Woolf [141].

[4]Hub-and-Spoke - A central point of integration for all the data coming from different sources, which then distributes the data further to consumer applications.

These can be implemented in both a P2P and a reusable manner. A discussion on how the two proposed styles and the work of Hohpe and Woolf (on middleware and application integration options) connect can be found in Appendix E.

The type of integration style, for example, P2P or reusable integrations will affect the overall interface numbers. If somebody uses a specific (P2P) integration this leads to an increase in the number of integrations which are part of a landscape. However, sometimes the use of P2P connections instead of reusable integrations might be needed. For example, if an application is storing sensitive data that should not be shared through a "reusable integration" or particular requirements are in place for only sharing data in a custom manner, then a P2P connection can be accepted. Furthermore, it can be the case that sometimes reusable or general interfaces are not needed, as one can wish to limit the applications that should access particular data.

As the interface choices affect the number of integrations (interfaces), the former does not have a direct link to application landscape complexity.

Application Functionality Overlaps

The relationships of functionality overlaps are explored next. During the survey and the interviews, it was mentioned that functionality overlap, defined as the number of applications delivering (some of) the same services, has one of the strongest links with complexity. If functionality overlap is solved, during the design phase, fewer data overlaps and integrations will be part of an IT landscape.

For example, if a specific client requests the usage of a technology which performs the same service as a similar application, the landscape complexity will increase. However, in the long term, the architect can choose to only add one of the two to the landscape. To illustrate this further, if an enterprise currently uses an application that stores and transmits particular data for a specific client, but another client does not want, can not use, or is not aware of this application, a request for a different one will be in place. As the enterprise needs to address and allow both clients to use the data, two applications performing the same tasks are needed in the current architecture. This example is illustrated in more detail in Chapter 5.

In case applications perform distinct tasks or services, data duplication becomes less likely. Furthermore, functionality overlaps drive the need for integrations, whereas integrations help with harmonising the application functionality. If there is a singular application performing a service fewer integrations will be part of the landscape.

Data Overlaps

The data overlaps refer to applications sharing/accessing similar or identical data entities (or common data sources).

To address this definition further, a master data management (MDM) view can be taken. MDM means *"envisioning how to organise key business information objects and govern their quality use and synchronization to optimize the use of information"* [153, p. 3]. In an ideal situation, one application/system should be the master of one data entity, as opposed to having the same data record being stored, shared and modified by several applications throughout the organisation without a master defined [154].

An example of this can be given by considering customer master data. In large IT landscapes, customer data is frequently stored in a myriad of systems each serving a different purpose which causes data issues, such as overlaps, duplication or replication. In this context, using an application as the master of the data is expected to create a point of reference which can reduce data overlaps.

Furthermore, when the same data is part of several application databases, creating common data models is expected to assist with reducing the complexity associated with data distribution or changes which can cause overlaps.

Given there will be data overlap, the number of integrations can increase, as more data consistency is needed. Moreover, it is believed that if one application could cover more data, then the number of P2P connections will decrease.

For creating a link with functionality, it is not always the case that if there are data overlaps more components will perform the same services.

For the overall complexity of the IT landscape, data overlaps cause an increase in complexity as data objects mastered in various systems will require more effort to change or adjust during a transformation process.

Number of Interfaces

Lastly, the main relationship of the number of interfaces is with complexity. Adding integration points which need to be actively managed to ensure applications communicate in an efficient manner, as well as understanding the processes of exposing and accessing data and functionality between applications are intricate endeavours.

4.3.2 Defined Factors

Based on Case Study

As the model seen in Figure 4.2 indicates that interfaces, data overlap and functionality overlap are the essential factors influencing application landscape complexity, each of them has been defined according to a similar template as the one proposed by Addicks and Gringel [155].

Given the outcomes of the survey and the interviews, it was discovered that the combination of these three is considered appropriate to initially quantify application landscape complexity. This is based on the responses of the participants involved in the case study. During the interviews, when their survey answers regarding the overall ranking of complexity factors were discussed, as seen in Figure 3.5, the three elements stood out as clear indicators for application landscape complexity. The rest were influencing these three elements, thus not directly associated with complexity.

Tables 4.1, 4.2 and 4.3 contain the definitions of each of the three indicators of application landscape complexity. Figures 4.3, 4.4 and 4.5 have also been created to illustrate the difference between a simple and a complex EA architecture, form the perspectives accumulated by the researcher.

Table 4.1: Application Functionality Overlap Definition

Name	Application Functionality Overlap
Description	Refers to the similar services delivered by the applications part of the landscape.
Use	Highlight the myriad of systems that perform similar tasks or provide comparable services.
Required Data	Application components linked with application services.
Metric	Count the number of application components delivering (some of) the same services.
Discussion	Can be gathered from ArchiMate models if applications are linked with services. This can be done by looping through services and verifying how many applications are realising an instance. It is believed that if the overlap in functionality increases, complexity increases linearly.

Table 4.2: Data Overlap Definition

Name	Data Overlap
Description	Sharing of similar or identical data entities (or common data sources).
Use	Showcases the intricacy of modifying data for an application which is present in numerous systems.
Required Data	Data sets are mapped to application components.
Metric	Count the number of applications per data set.
Discussion	Can be gathered from ArchiMate models by leveraging a data object. Analyse how many applications are accessing the data objects. Complexity increases exponentially due to more data transformations, dependencies etc.

Table 4.3: Interfaces Definition

Name	Interfaces
Description	Points of interaction between different application landscape components.
Use	Show the process of exposing and accessing data and functionality between applications.

Continued on next page

Table 4.3: Interfaces Definition (Continued)

Name	Interfaces
Required Data	Accurate and actual representation of the number of interfaces.
Metric	Count the number of interfaces.
Discussion	Can be gathered from ArchiMate by counting the number of data flows. In other cases, if organisations are using integrations as a component, the overall number can be counted. Each new interface adds additional complexity, and the increase is exponential.

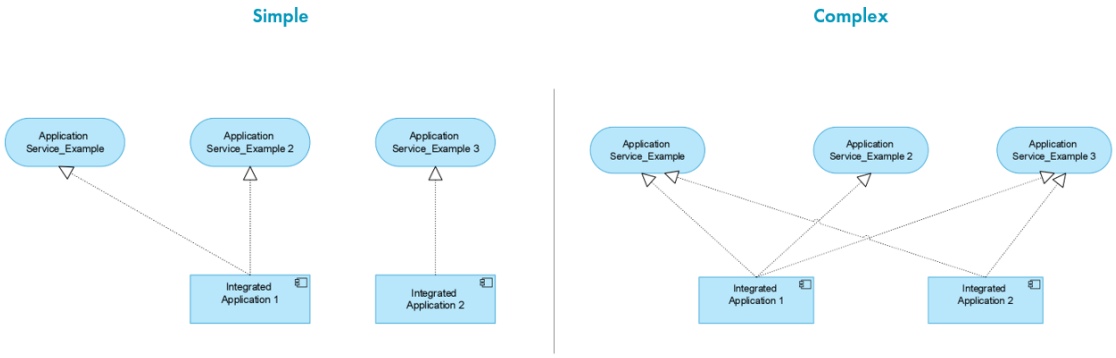


FIGURE 4.3: Application Functionality Overlaps - Simple vs. Complex Architecture

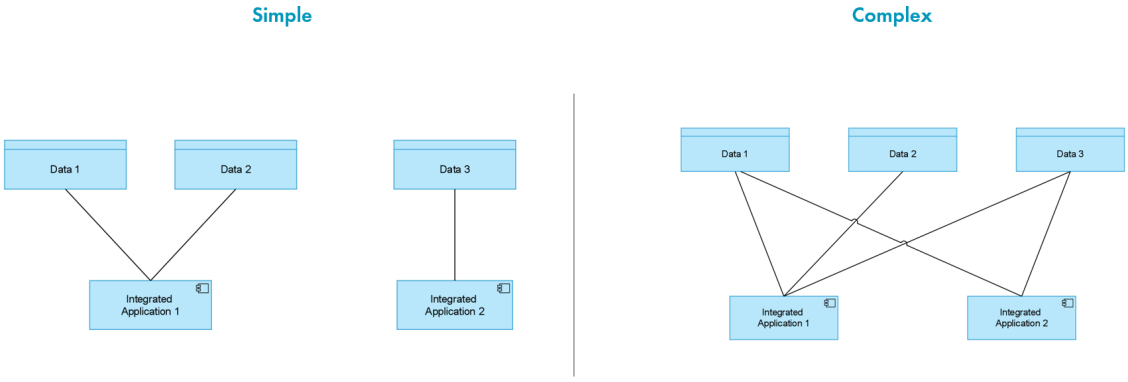


FIGURE 4.4: Data Overlaps - Simple vs. Complex Architecture

Discussion Points

These three figures and tables (Figures 4.3 to 4.5 and tables 4.1 to 4.3) have been modelled based on the case study outcomes, hence, it is also essential to discuss their potential links with the literature.

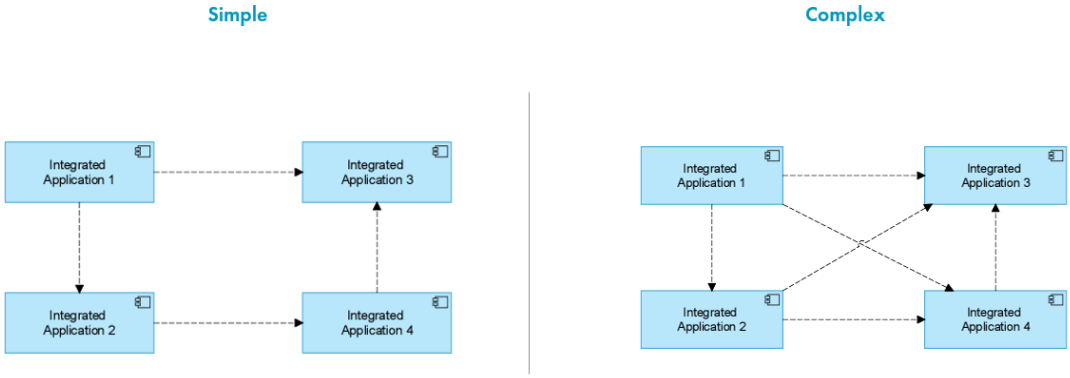


FIGURE 4.5: Interfaces - Simple vs. Complex Architecture

Table Contents

Starting with the tables, for each, several points need further clarification. For example, in the case of application functionality overlaps, the required data is "application components linked with services". This means, that an organisation must know what service each application realizes. This link, for example, can be represented as a realization relationship in ArchiMate between an application component and an application service.

Semantics is another issue that can impose difficulties in understanding similarities of applications in information flow exchanges or types of offered services. To acknowledge which data entities or services are the same when looking at the application landscape, the terminology across different departments must be aligned.

As discipline is required, the tables introduced should be tailored to a specific organisation to match their current way of modelling application landscapes and support the discussion on how this can be improved (e.g., ensure the use of the same terminology).

Figures' Views and Patterns

When considering the figures, these are designed to match the proposed viewpoints in ArchiMate. A viewpoint, in a modelling language such as ArchiMate, represents a relevant subset of elements and their relationships [1]. The types of viewpoints under which the three figures can fall are given in the list below [5]. It can be observed that these views are all part of ArchiMate's application layer, matching the main objective of the thesis to quantify ALC.

- Figure 4.3 which depicts functionality overlap is an Application Behaviour View. This describes the "internal behaviour of an application as it realizes one or more application services" [1].
- For data overlaps, in Figure 4.4 an Application Cooperation View can be observed. By modelling this, it is observed how applications are using data, and what information flows between them.
- For the number of interfaces, Figure 4.5 depicts an Application Cooperation View. This type of view describes the information flows between applications or looks at

[5]For more information on the viewpoints, their definitions, and examples one can check the following ArchiMate specifications - https://pubs.opengroup.org/architecture/archimate2-doc/chap08.html

the services they offer and use. "This view is typically used to create an overview of the application landscape of an organization" [1].

Furthermore, in the figures, some architectural patterns can be observed. These are ways of putting building blocks together for different IT contexts and scenarios. Four patterns are put forward by Britton and Bye [156]. For the functionality overlap, the difference between single-centralized and multiple-centralized patterns can be observed. In a simple context, one application is responsible for one service (single-centralized), whereas a complex landscape is commonly associated with a multiple-centralized pattern as several applications help realize the same service.

When considering working with data overlaps, the difference between a pass-through and a copy-in/out architecture pattern should be considered. Pass-through would mean that applications are passing data from one to the other (relying on each other). Due to timing dependencies issues with data being replicated, stored twice etc. can emerge. The copy-in/out pattern is associated with creating a common place for storing and distributing data, e.g., using a data lake. In this case, data overlaps can occur if the same applications are working on the same data source at the same time.

4.4 Measurement Method

4.4.1 Requirements

This section offers an overview of the requirements for a method to quantify application landscape complexity, as described by the participants in the case study. Moreover, it details the choices behind the artefact design.

Formula for Quantifying Complexity

For designing the formula that can calculate the complexity of a solution option, the three factors (application functionality overlap, data overlap and the number of interfaces) are initially considered to be a sufficient combination indicating ALC. As mentioned previously, these three were observed during the case study as having direct links with the growth or the decrease of complexity, whereas the other drivers and indicators of complexity affect these three. This aligns with one of the requirements collected during the case study, that of first having a few easily understandable complexity indicators that can be quantified. The definitions and the designs introduced in the section above (simple versus complex architecture), also support the architects.

Moreover, as the formula is required to give a sense of direction, the delta in complexity can be calculated. This means subtracting the complexity score of an as-is situation from the one of a to-be. By having this delta score, an IT architect can faster see which transformation of the application landscape is simpler.

Method for Embedding Formula

Several other requirements were gathered during the interviews. One was to generate scores based on EA designs. Another was making it very straightforward as to what needs to be inputted into the formula. The next requirement was to create a feeling of direction by using visual cues. Considering these points, Excel has been selected as the platform to support the initial method development.

4.4.2 Design Choices

Formula Choices

As seen in the literature, the work of Daoudi [66] calculates complexity by summing indicators which are changing from one state to another (from a period "t", to a future state at time "t+1"). Additionally, other works such as [4], [157], or [158] highlight the importance of assigning weights to the indicators in a complexity formula. Hence, the proposed formula for ALC is a mix of these two directions. The ALC uses a summation of indicators of complexity, each having its individual weight.

The proposed formula for application landscape complexity is introduced in equation 4.1. For an absolute value of complexity, the number of applications delivering the same or some of the same services needs to be counted, alongside the number of applications using the same data sets and the number of interfaces. By summing them and assigning weights, the result of the formula can be used as an indicator to guide architects in making choices for the future

$$\begin{gathered} \text{Application Landscape Complexity (ALC)} = \sum_{i=1}^{n} \text{Count}_i \times w_i \\ \text{where } n \in \{F, D, I\} \\ w_F = 0.2 \quad w_D = 0.3 \quad w_I = 0.5 \end{gathered} \tag{4.1}$$

The initial iteration of the formula should be kept simple and adjusted in later stages based on project solution options examples within a company. The initial weights to be used for the three elements are 0.2 (functionality overlap), 0.3 (data overlap) and 0.5 (interfaces). The first criterion behind these chosen weights is the outcome of the ranking during the survey. In the ranking, it was visible that respondents consider the number of interfaces a more important indicator of complexity, followed by data and application functionality overlaps.

The second criterion used is the level of difficulty in making certain choices to avoid complexity. For example, functionality overlap is, in most cases, a design choice which can easily be adjusted, whereas having a large number of interfaces, makes it hard to control how a change in one ripples across the landscape. For data overlaps, adjustments were considered harder to control than working with application functionality overlaps, but easier to understand as opposed to grasping the changes in interfaces. These points emerged after several discussions performed during the case study.

To picture the weights in a context, an example of an IT architecture of a company can be delineated. In the application landscape, there is rarely any middleware, hence a lot of point-to-point connections between applications are visible. In the case of multiple P2P connections in an architecture, more applications are believed to be using the same data entities. Hence, issues such as data overlaps can emerge. Furthermore, in this landscape, the number of applications performing similar services increases if the architects are making a deliberate choice to add new ones. Based on this example, it can be observed how the three weights align with the landscape choices in this situation.

Once the as-is, and the solution options are assigned a score, the delta in complexity will be calculated, to give the architects a clear indication of which direction is considered

simpler. This will be done by calculating the difference between the target state and the initial state's complexity as seen in equation 4.2.

$$\triangle \text{ ALC } = \text{ALC}\, target - \text{ALC}\, initial \tag{4.2}$$

Method Choices

To embed the formula and use it for solution option choices, an Excel tool was created. A GitHub repository[6], contains the spreadsheet with the application landscape formula. This is an accepted initial stage (according to the company at which the case study was conducted) to quantify complexity. The tool includes a table with the indicators for complexity and space assigned for architects to fill the necessary values per option. Once these are filled in, two generated scores will be visible: the ALC for a singular view and the delta in the change. Figure 4.6 shows this table. The delta in change also uses colours as a visual cue that helps an architect observe which solution has a low (green), medium (yellow) or high (red) complexity score.

Furthermore, a definition of the three indicators is provided, such that architects will fill in the tool according to set guidelines. Figures 4.7, 4.8 include a snapshot of such definitions. The first columns include textual comments and the last two include a guide as to what a simple or a complex indicator might look like if designed in a modelling language such as ArchiMate.

Quantifying Application Landscape Complexity

Project Analysis

		Necessary Input	As-Is Situation	Solution Option 1	Solution Option 2	Solution Option 3
Indicators	Application Functionality Overlap	Count the number of applications performing (some of) the same tasks/services	2	2	2	2
	Data Overlap	Count the number of applications per data set	2	2	2	2
	Interfaces	Count the number of interfaces	3	5	7	5
Complexity Scores	Application Landscape Complexity	Calculation of Application Landscape Complexity	2,5	3,5	4,5	3,5
	Delta of Change	Difficulty of change	0	1	2	1

FIGURE 4.6: Fill in Table

4.4.3 Procedure

Based on the model and the formula for quantifying application landscape complexity at an absolute level, the following steps must be performed by architects when evaluating project solution options. First, the project options need to have designs illustrating what is being changed. Next, based on these designs, the three indicators in the Excel file need to be filled in. Lastly, an informed choice can be made based on the scores generated.

The next Chapter, 5, offers an in-depth example of how the method can be applied to an example from the company at which the case study was conducted.

[6]GitHub - developer platform. The researcher can be contacted for collaborator rights/access to the private GitHub or future ideas for tool development. Alternative link: https://gitfront.io/r/evastoica/nX7kKTGXTAYX/ALC-Quantification/

Definitions						
		Description	Use	Required Data	Ideal Case	Discussion Points/Exceptions
dicators	Application Functionality Overlap	Refers to the redundant or similar features shared across applications which are part of the Application Landscape	Highlight the myriad of systems that perform similar tasks or provide comparable services	Application components linked with services	Can be gathered from Bizzdesign if applications are linked with services	Overlap in functionality increases, complexity increases linearly - weight 0.2
	Data Overlap	Sharing of similar or identical data entities, or common data sources	Showcases the intricacy of modifying data for an application which is present in numeorus systems	Data entities are mapped to application components	Can be gathered from Bizzdesign by oberving how application components are associated with data objects	Complexity increases exponentially due to more data transformations, dependencies etc.- weight 0.3

FIGURE 4.7: Definition Table - 1

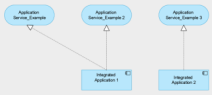
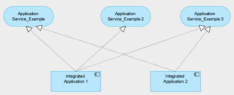
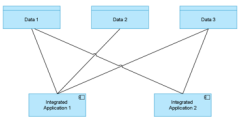
Required Data	Ideal Case	Discussion Points/Exceptions	Simple Design	Complex Design
Application components linked with services	Can be gathered from Bizzdesign if applications are linked with services	Overlap in functionality increases, complexity increases linearly - weight 0.2		
Data entities are mapped to application components	Can be gathered from Bizzdesign by oberving how application components are associated with data objects	Complexity increases exponentially due to more data transformations, dependencies etc.- weight 0.3		

FIGURE 4.8: Definition Table - 2

4.5 Answers to Research Questions

Prior to diving into the validation of the proposed artefact that can assist architects with the management of EA complexity and improve their agility when responding to change, it is important to detail the answers to the third and fourth research questions investigated in this research.

4.5.1 Complexity Metrics

The third question, "What are suitable complexity metrics that can be used in measuring the delta of the change in the application landscape?", can be answered based on the model proposed in Figure 4.2. From a myriad of indicators, the architects consider that a combination of three of them can adequately indicate the level of complexity. These are application functionality overlap, the data overlap and the number of interfaces. These are measured by simple counts. By having an understanding of the level of ALC for a certain state, one can calculate the difficulty of change by subtracting the complexity of the initial state from the one of the future state.

4.5.2 Cost of Complexity

The application landscape complexity is also expected to drive certain costs in an enterprise, as discussed in Section 3.4.2 (based on the case study outcomes) or as visible in

Figure 2.3. To respond to the fourth research question "What costs can be factored in a formula when application landscape complexity hinders the agility to adapt to a change?", and combine the literature and the case study insights, the model introduced in Figure 4.2 is expanded with a cost column, as seen in Figure 4.9.

Normally, the project costs or the run and maintain costs (e.g., licenses, infrastructure, training, labor, interface build cost etc.) are being considered when choosing between options or potential target states. These are discussed both in literature and in practice. Nonetheless, as an application landscape can be complex, several "hidden" or "invisible costs" are not evaluated. It is not yet possible to quantify the costs incurred by complexity. This means that no formula can translate a complexity score into a monetary value. An initial proposal would be for the cost of complexity to be calculated as the sum of these invisible costs.

These types of costs stream from missed opportunities and available resources to prolonged decisions or reworks. Table 4.4, has been designed to explain the "hidden" costs of complexity. By having this monetary value organisations can have a better sense of direction. To create a further connection between costs and complexity, based on the monetary value, the ALC formula can be calibrated in a certain manner.

Table 4.4: Costs of Complexity Breakdown

Type of Cost	Example	Mentions (Literature [ref nr.] or Case Study [C])
Missed opportunities	Refers to what can be incurred by a company which has a complex landscape and can not adjust quickly to changes. e.g., In an IT landscape which uses numerous legacy applications, adding newer applications which might be better fitted can become a problem. Opportunities which can lead to a higher revenue might not be taken as the landscape does not permit adaptation.	[C], [119]
Cost of available resources	To handle a complex application landscape, a specific set of skills or certain technology components might be needed in the future. e.g., If an application uses an outdated programming language or has a high level of customisation, acquiring the people who can work and maintain it can prove to be very costly.	[C], [112]

Continued on next page

Table 4.4: Costs of Complexity Breakdown (Continued)

Type of Cost	Example	Mentions (Literature [ref nr.] or Case Study [C])
Cost of analysis	If an application landscape is complex, more time will be spent by an IT architect in understanding the overall architecture, gathering insights about the works of the systems and components etc.	[C]
Cost of prolonged decision processes	In a complex application landscape reaching a decision is not a straightforward process. In this case, more time is spent by various people (IT architects, managers etc.) to reach an agreement on the best way forward. Here, the costs are the man-hour costs spent on making decisions.	[C]
Cost of knowledge management	If one organisation is having a highly complex IT landscape, tools to keep track of all the changes and the knowledge of the people who piece together the landscape are needed. This means a higher cost of acquiring systems that can manage the knowledge within the organisation. According to one respondent, "Everything becomes complex and you need to build systems to keep track of knowledge".	[C]
Additional implementation of subsequent change	This cost refers to the money spent on the subsequent changes made after a project is completed. For example, if a project implements a decision which only increases the complexity of an IT landscape, more money is expected to be spent in the future for the landscape.	[C] , [106]
Cost of rework	When dealing with ALC, architects spend more time in fixing issues, such as functionality overlaps or data overlaps.	[C], [92], [110], [111]

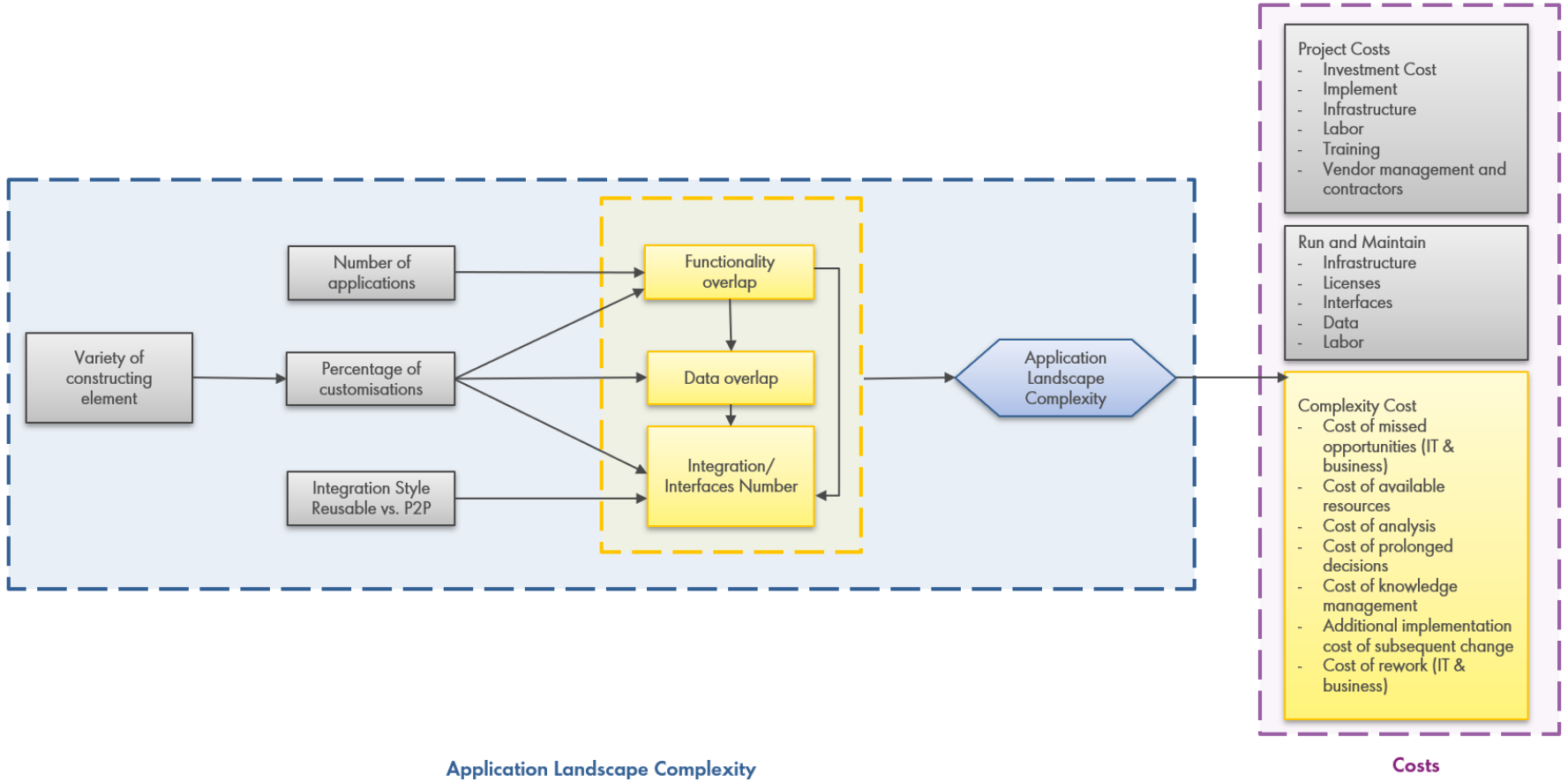


FIGURE 4.9: Costs Driven by ALC

4.6 Summary and Takeaways

In Chapter 4, the details behind the design of the artefact are illustrated. By first understanding the methodology that served as a basis for the process, one can follow the logical structure used to arrive at the proposed measurement method for application landscape complexity. The most important points of this chapter are synthesised in the list below.

- The artefact aims at measuring application landscape complexity based on documented solution options designs to help in effective decision-making.
- By creating a measurement model, it became apparent that there are three sufficient indicators for ALC. These are the number of interfaces, the application functionality and the data overlaps. They need to be clearly defined to align with the architectural styles used by an enterprise.
- A formula for ALC is seen in equation 4.1. This is a sum of the counts of indicators of complexity multiplied by certain pre-assigned weights. This can be further calibrated by using the "hidden costs" (e.g., missed opportunities).
- When trying to calculate the cost of complexity, after assessing the ALC score of a state, given it is complex numerous activities need to be done by an organisation to manage complexity. The cost of these "hidden costs", such as prolonged decisions or finding the right resources can be summed to indicate the cost of complexity.
- A formula for complexity should give a feeling of direction, hence using the delta in application landscape complexity can support architects in understanding which end state has a lower ALC score.

Chapter 5

Treatment Validation

Chapter 5 moves to the third step of the Design Science Research Methodology, the validation phase. In the first section of this chapter, 5.1, the procedure followed for the validation of the proposed artefact is introduced. Next, Section 5.2, exemplifies how the method for quantifying application landscape complexity can be used. This is applied to a documented solution option from the company at which the case study was conducted. Sections 5.3 and 5.4 detail the validation of the formula and its application.

The chapter ends with a proposal for the automation of the formula for quantifying application landscape complexity, in Section 5.5, and a summary of the topics described in this chapter, in Section 5.6.

5.1 Procedure for Validation

This section depicts the questions based on which the validation procedure ran. A discussion on the types of validation used for this research is then introduced.

5.1.1 Validation Questions

For the validation of the treatment, both the formula and the tool were assessed following the principles and guidelines proposed by the DSRM methodology. Wieringa [67] proposes a list of questions for artefact validation. For the thesis, the following questions guided the validation process.

1. Effects and Requirements - *Artefact X Context = Effects*
   (a) How does the artefact (tool and method) help improve the management of complexity?
   (b) To what extent can the artefact be applied to solution option choices in projects?
   (c) How can the artefact be embedded in projects?
   (d) What needs to be done to embed the artefact [in projects/in the company]?
   (e) Does it [the artefact] assess complexity at a quantitative level?
   (f) Does it [the artefact] bring value to the work of IT architects? (e.g., help support during decision-making processes)

(g) Can the Excel tool be fully filled in based on documented solution options?

(h) Can the identified cost of complexity be factored in the ALC formula?

2. Sensitivity - *Artefact X Alternative Context = Effects*

   (a) How can the tool be used in a different context? (e.g., not applied in projects)

   (b) What assumptions need to be in place for the use of the tool in different contexts?

3. Trade-offs - *Alternative artefact X Context = Effects*

   (a) What are other approaches/tools (if any) that can help with quantifying application landscape complexity?

   (b) How does the Excel tool perform as opposed to alternative artefacts?

5.1.2 Validation Types

Based on these questions, two types of validations were used. The first is expert validation, whereas the second is technical action research. Expert validation is concerned with the design of treatment and how this can interact with envisioned contexts. Then, with technical action research one could apply the artefact to a real-life problem and help a stakeholder learn from this. By leveraging both expert opinions and technical action research, sufficient evidence can be collected that the proposed method will support architects with quantifying ALC and assist with the management of complexity. This is an important element described by Gerike et al. [149].

The design of the artefact (the Excel tool, as well as the method) was submitted to a panel of experts who envisioned how it would help with the management of complexity or if it met the initial requirements, described in Chapter 4.

In the process of validation, first, the measurement model was introduced alongside the formula for ALC. Then, the method of how to quantify complexity was detailed based on a demonstration of the tool as a support for solution option choices during projects (see Section 5.2 for the example). Then, the expert panel was asked the set of questions proposed above. Furthermore, these findings about the model, equation for complexity and steps to apply the equation were also shared with the interview group, who could also provide answers to the questions on requirements, effects, sensitivity and trade-offs.

Two different types of outcomes from the expert panel were collected: one concerned with the equation for calculating complexity and one focused on the method applicability. These outcomes are detailed in Sections 5.3 and 5.4.

Further considerations, such as validity, inference, generalisation, interpretation and problem anticipation, as proposed by Richey and Klein [128] have been taken into account for a more thorough validation process. For the validity of the tool use, experts with different areas of specialisation participated in the validation process. The main validation process took place as a focus group, in which four IT architects working in domains such as innovation, cross-business or asset management responded to the questions introduced above. These were the architects part of the "advisory group". Furthermore, five other participants (three architects and two employees in a management position) provided their insights in discussions on the tool's validity.

As for inference, the artefact's practicality and effectiveness have been investigated. This has been done with the help of TAR, given that one of the architects was sent the Excel file. Based on the project that was currently in progress, the architect could fill in the tool and mention how he perceived the measurement method and if new insights were learned from this. The constraints on the tool, the plan of use, the usability analysis and expected problems have also been discussed with the experts.

5.2 Method Application Example

An example is given to illustrate how the tool can be applied in organisations, based on EA models. The organisations which can use the measurement method should have an application landscape with at least several connected systems realising services and exchanging data to support the business. Moreover, the organisations should have projects running for which EA models of the as-is situation and possible to-be architectures are documented. The other characteristics of organisations should not impact the ability to use the measurement model.

The example is created by generalising a project conducted at the company involved in the case study. The project is concerned with bringing geospatial data to a Digital Twin (DT) for monitoring active refinery leaks. In the company case, the DT represents the digital model of different equipment from refineries (for example pipes through which substances such as oil, water, gas etc. flow), which can help with monitoring or maintenance, and ultimately, by combining various datasets, assist in creating a full situational awareness for employees. The equipment is supported by a myriad of applications which should be connected to provide insights for the DT. These interconnections and potential overlaps in application functionalities can lead to a higher application landscape complexity.

Presently, refineries are using two solutions for active leak data, which are delivering the same services. One is the desired architecture decided by both business and IT, whereas the other is an intermediary solution part of the landscape due to the client's requirements. Hence two applications use the same data, the one about the leak, and perform the same service, which is to monitor and provide this data to other systems. These will be called Application A (the one desired by the company and used by certain clients now) and Application B (the one requested by a specific client).

In the current situation, the as-is, only Application A is connected to the data platform and provides information to the DT. The data platform is an application which acts like a hub-and-spoke model, allowing the organisation to manage all the interfaces in a single place. Application B appears as two instances but is counted as one component. The as-is state is represented in Figure 5.1.

For this project, several options were considered on how to adequately bring the data from Application B to the DT, by either using a data platform or creating point-to-point connections.

Option 1, as seen in Figure 5.2, connects Application B to the data platform, hence increasing the number of interfaces. Next, option 2, depicted in Figure 5.3, explores the connection of Application B to both the data platform, as well as to the DT via point-to-point connections. This solution is evaluated as real-time data requirements might be in place and the data platform might not be able to meet them. Lastly, the third option, Figure 5.4, integrates Application B with the digital twin via P2P connections. The P2P

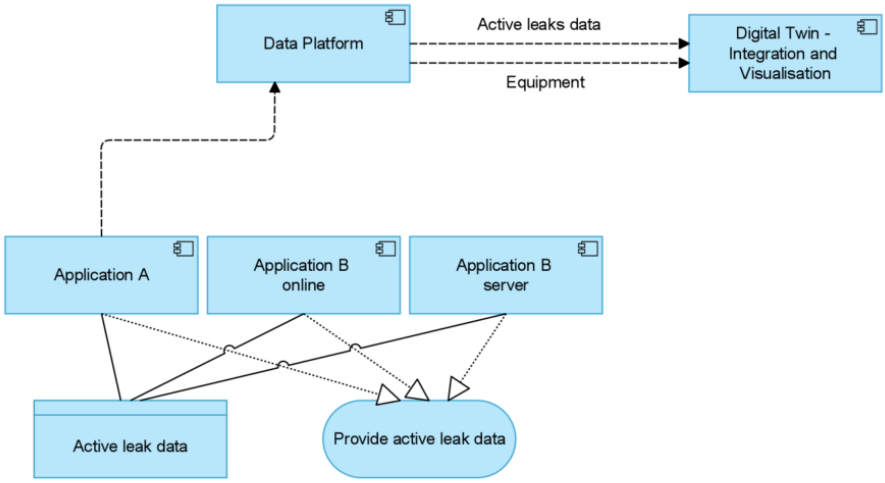


FIGURE 5.1: As-Is - Digital Twin

connection in this situation refers to tightly coupling Application B with the DT, without the use of any middleware.

For this example, based on the documented solution options, the IT architect responsible for making the changes was asked to fill in the Excel according to the predefined template (as introduced in Chapter 4 and further exemplified in this section). Hence, the architect was asked to count the number of interfaces and the application and functionality data overlaps in the as-is and to-be states of the project.

The appropriate cells in the Excel file had to be filled in with these numbers. Figure 4.6 contains the overview of the assigned scores by the IT architect. The as-is and all three to-be options have 2 applications performing the same functions, as well as 2 applications with data overlaps. What changes in each of the to-be situations is the number of interfaces. The formula indicates that Option 1 and 3 are simpler than Option 2. The outcome aligns with the general view of the architect.

As Option 1 and 3 have the same ALC score (3,5), when considering agility and making a change in the future, the architect's choice was influenced by the type of integration used. Option 3 creates a P2P (specialised, not reusable) connection and does not make use of the data platform. This is perceived as more complex, hence Option 1 was selected.

This decision point showcased one potential point of improvement for the formula and method. After calculating the delta in the change of complexity for each option, in case of equal scores or particular requirements, adjustments to the formula can be made by factoring in the: integration style, percentage of customisation, number of components in the landscape or the variety of constructing elements.

As in this scenario, the integration style played a relevant role, the ALC score can be adjusted. For example, in the future, based on multiple project analyses, the fact that a solution used a reusable integration as opposed to tightly-coupling components via P2P (specialised) connections, can result in a deduction of a certain score from the overall complexity score. To illustrate this, as Option 1 used middleware which supports the reusability of the interface in the future, the delta in change decreased with a value of 0.2.

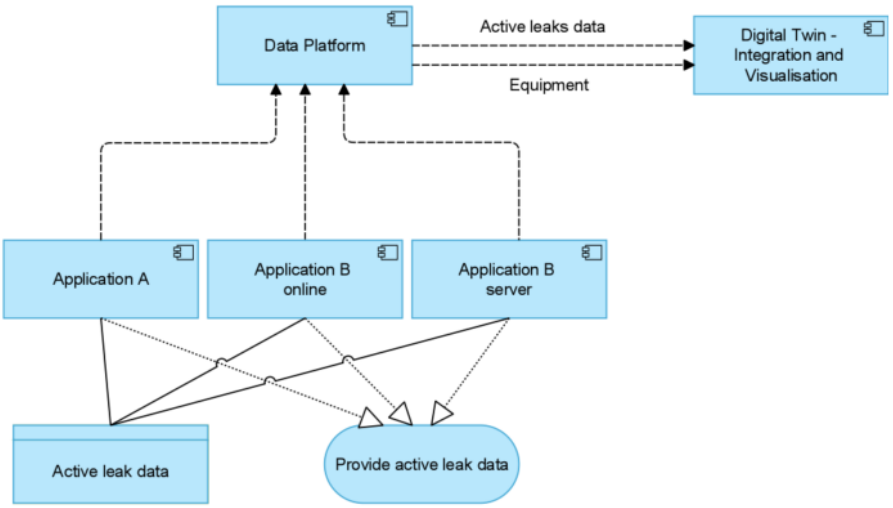


FIGURE 5.2: Option 1 - Digital Twin

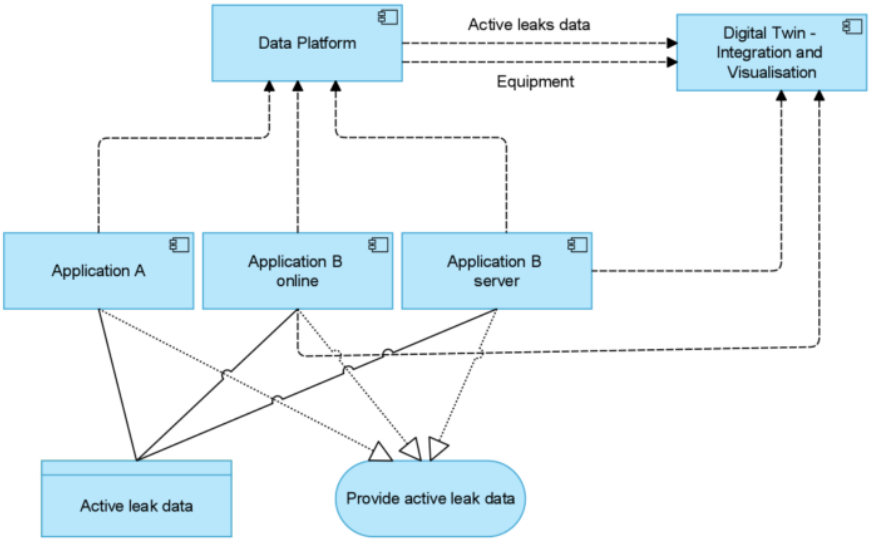


FIGURE 5.3: Option 2 - Digital Twin

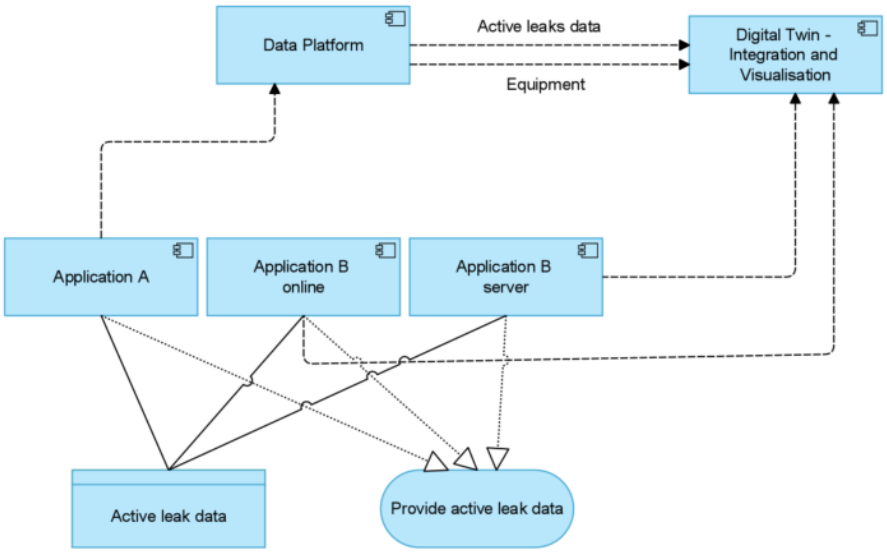


FIGURE 5.4: Option 3 - Digital Twin

In Figure 5.5, it can be seen that the type of integration was mentioned per solution. As such, the score in the delta of complexity was changed, as per Figure 5.6. Now it becomes clear that Option 1 is a less complex choice.

Additional Indicators

Name	Solution Option 1	Solution Option 2	Solution Option 3
Number of Applications	...	...	...
Integration style	Reusable	Both	Point-to-Point
Percentage of Customization	...	...	...
Variety of constructing element	...	...	...

*In this case study the type of integration influenced the choice ; these factors need to be explored in later iterations for adjustings the complexity scores

FIGURE 5.5: Adjustments to ALC Score - Type of Integration

As a general remark, if in the future the organisation chooses to only use Application A, as seen in Figure 5.7, the complexity score will decrease (and its value would be 1,5) as there will be no more data or functionality overlaps. This option is only introduced by the researcher, as it was not explicitly mentioned during the case study.

5.3 Formula Validation

Starting with the formula validation, several points for improvement were brought to light during the discussion with the experts. These concern the initial weights assigned to the indicators (application functionality overlaps, data overlaps and the number of interfaces), the possibility of introducing powers given the influence of the indicator on complexity or adjusting the score of complexity based on additional factors. These suggestions are collected from a limited group of respondents, therefore they do not represent the best way forward, rather they are just stirring discussions for formula improvements.

		Necessary Input	As-Is Situation	Solution Option 1	Solution Option 2	Solution Option 3
Indicators	Application Functionality Overlap	Count the number of applications performing (some of) the same	2	2	2	2
	Data Overlap	Count the number of applications per data set	2	2	2	2
	Interfaces	Count the number of interfaces	3	5	7	5
Complexity Scores	Application Landscape Complexity	Calculation of Application Landscape Complexity	2,5	3,5	4,5	3,5
	Delta of Change	Difficulty of change	0	1	2	1
Adjusted Score				0,8	2	1

FIGURE 5.6: Change in the Delta of ALC

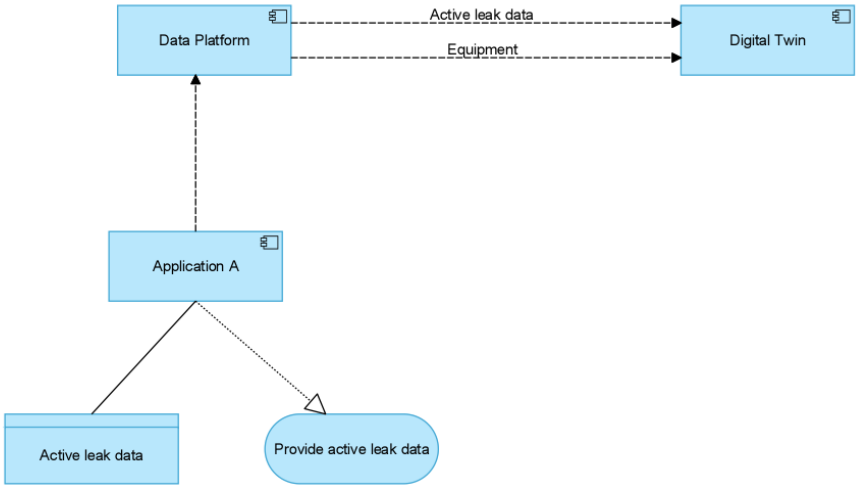


FIGURE 5.7: Option 4 - Digital Twin

5.3.1 Weights

The initial formula is considered a sufficient starting point for quantifying application landscape complexity at an absolute level. This is the case, as the experts think it can adequately indicate complexity by using simple, easy-to-understand and collect indicators and metrics. This simplicity is essential for a measurement method that must be applied in practice in the organisation. By starting to test it in several cases, the initially assigned weights can however change.

One suggestion received from the expert panel was to fit the weights according to business situations or dynamics (e.g., number of users), or style of desired solutions. For example, in an organisation which uses microservices, maybe the weight for the interface number is lower than the other two. Hence, flexibility (to a certain extent) needs to be given to an enterprise when using the formula. It is important to note that the weights must not change for individual projects or portfolios as this will generate skewed and inconsistent results. A more detailed discussion regarding weight adjustments is part of Chapter 6.

This aspect of flexibility is also connected to trying to keep the formula's result as close as possible to an absolute indicator. In this case, it is important to not try and intertwine or factor in aspects such as relative assessments or severity of complexity. The process of quantifying complexity should not look at drivers of complexity, calculate a score and then offer a good or bad, appropriate or inappropriate assessment/judgement (e.g., a high complexity score is bad). Rather it should just mention the score for ALC for the comparison of designs (e.g., between options choose the one with a lower level of complexity).

This type of judgement of whether a high ALC score indicates a good or a bad change is a future work (a more in-depth explanation is part of Chapter 7.4). By applying the formula in several cases, the architects can start connecting the score to understand the meaning of the changes resulting in it and if potential benefits arise with a lower ALC score.

5.3.2 Powers

Another mentioned addition to the formula was including powers for each of the three indicators. Each of them can influence complexity in various ways (e.g., linearly, exponentially), thus factoring in how a change in one influences the application landscape complexity would be interesting to explore. In literature, Rojas [97] proposed a formula based on which the functionality, data overlaps and the number of interfaces could all be assigned the power 3.11. However, the discussions with the panel of experts revealed that this is too high and there should be a clear differentiation between the powers of each indicator.

Presently, having powers is not perceived as a necessity as "it overcomplicates the formula", as one of the experts mentioned. In the case of observing the need for powers after multiple tests conducted with the initially proposed formula, these can be included.

Another important distinction is that when including powers in the ALC formula, the weights assigned to each indicator need to be discarded, as mentioned by experts. So the complexity formula should not contain both weights and powers. The experts consider both weights and powers to indicate the impact of the indicator on complexity, hence creating a duplicate score. From a mathematical perspective, the weight scales the magnitude of the indicator, whereas the power shows the growth rate.

As a mathematical implication, this means double counting [159] due to the overlap of effects. When a weight represents the impact of an indicator on complexity and the power shows the growth of the indicator due to complexity, they both capture the connection between indicators and complexity. There will be a risk of double counting the same effect and overestimating the complexity score.

If one uses to wish both powers and weights in the ALC formula parameter tuning or calibration is needed to find the right combination of values that can yield generalisable results.

5.3.3 Indicators

Lastly, to embed the formula in projects, indicator-related suggestions were received. The formula can be improved considering the other four factors in the model: integration style, the percentage of customisations, the number of applications and the variety of the constructing element. First, each of the four indicators needs to be described.

For example, the integration style is detailed in Table 5.1 and Figure 5.8 (for both the

information in the table and the simple and complex view, a discussion similar to the points mentioned in 4.3.2 can be held). Next, if one solution uses reusable interfaces or middleware, then complexity can decrease as opposed to using P2P connections (specific interfaces) between applications. This type of reasoning will be needed per each remaining indicator.

During the validation procedure, feedback suggested that the preliminary formula (the one limited to the use of three indicators) can be adjusted. This meant subtracting a predetermined weight if, for example, the solution adds a reusable interface as opposed to a P2P connection, or if there are more customisations in the landscape. However, after further expert analysis, it was mentioned that the other four factors can also use a similar formula structure as the three simple ones. This means that in more advanced iterations of the formula, the complexity indicators can have the same structure, e.g., a combination of counts and weights, which will either be added or subtracted to create the overall ALC score.

In Chapter 4 the integration style choice is mentioned. For the research, two disjoint categories of integrations are chosen: P2P connections and reusable integrations.

For initial usages of the formula, once again, it was stated that factoring these four indicators is not necessary. This is the case as it is hard to derive weights for each of the four factors, adjust the scores, or collect information about all of them from an enterprise.

Table 5.1: Integration Style Definition

Name	Integration Style
Description	Different modalities of connecting application components for achieving a certain functionality.
Use	Illustrates how the connection between applications is realised.
Required Data	Clearly defined properties/specifications of an interface.
Metric	Check if interfaces are reusable or P2P.
Discussion	Can be gathered from ArchiMate by considering the flow properties. If a reusable integration is used as opposed to a P2P connection complexity decreases.

Another avenue of exploration mentioned by the experts was creating more flexibility per indicator. For example, instead of counting the number of applications delivering the same service, maybe consider the applications performing certain functions or assisting with certain capabilities. This recommendation will not be taken into account as it can create inconsistencies due to different levels of observation and subjectivity.

The costs driven by the application landscape complexity, as displayed in Figure 4.9 should be summed together to offer a monetary value for the cost of complexity. When this sum is determined or can be estimated, the suggestion to calibrate the ALC score was made by one of the respondents. The monetary value can be used to adjust the weights and powers of the three indicators, such that these can represent reality.

Figure 5.9 offers an overview of all the possible changes to the initially proposed formula for ALC, as shown in equation 4.1.

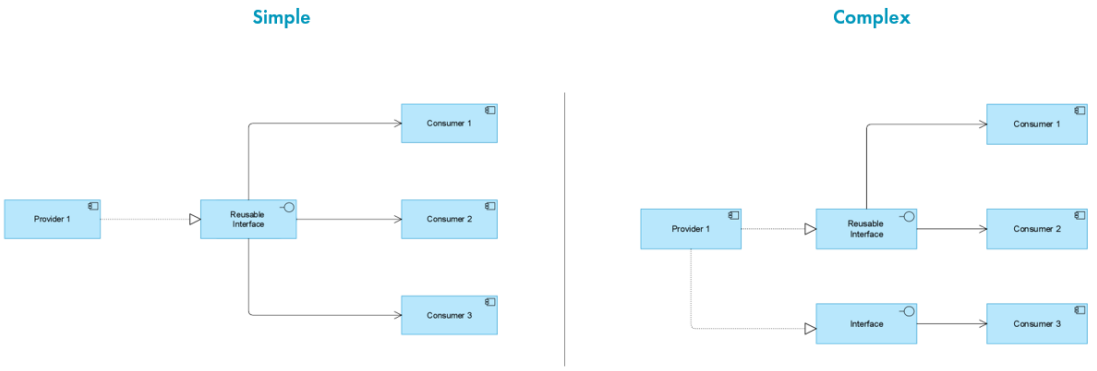


FIGURE 5.8: Integration Style - Simple vs. Complex Architecture

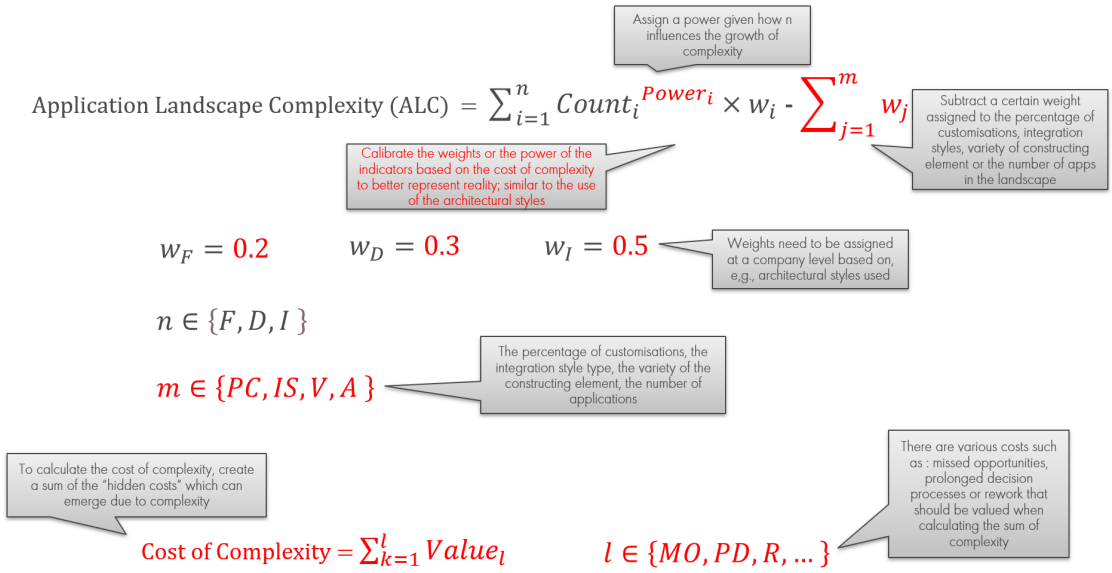


FIGURE 5.9: Formula Suggestions

5.4 Method Validation

Going beyond the simple formula validation, the second factor under investigation was the measurement method. For the research, it is essential to uncover if this method can be applied during projects to help architects manage complexity and make effective decisions.

The experts agreed that the method can be used by IT architects and that it helps as a solution option tool for the management of complexity. By utilising the proposed artefact, application landscape complexity is quantified alongside the delta in the change from an as-is to a to-be situation. The addition of the definitions and a clear design of what complexity would mean for each indicator is expected to make the tool easy to use and understand. Hence, an option choice can be made based on the generated score of complexity.

In projects, options are chosen based on a wide array of elements, such as strategic, functional, or non-functional fit. A list with these choice points has been created after analysing documented EA solution options during the case study period. The left column in Figure 5.10 depicts these points which are influencing or guiding decisions. As the complexity

of the application landscape is rarely considered (the model in blue in Figure 5.10 - or in the middle), the artefact is expected to help the enterprise. Furthermore, for solution options, costs that influence a choice are mostly run and maintain or high-level project costs, as seen in the purple dotted square (right column in 5.10). Factoring in the costs of complexity will bring value and allow the architects to make better-informed choices.

It is expected that the artefact can be applied to various contexts, going beyond project solution option choices. For example, the experts are seeing it helping at a portfolio level or a line of business level. In the case of portfolios, the artefact is expected to become a strategic portfolio analysis tool that can indicate where the highest complexity streams from. The outcome of the formula is believed to "provide input for a governance process between portfolio managers, IT managers and lead architects" (as mentioned by one of the experts). This would be a first step based on which, the business can start rationalising or creating a roadmap for the future. Additionally, when working with different clients, the artefact can also be generalised to indicate the level of complexity associated with their application landscapes.

For trade-offs, doing the measurement in Excel is considered "a good first step" by the experts. At the moment, the necessary discipline for automation in tools such as PowerBI [7] or Bizzdesign needs to be improved. Even if other tools exist to help with the management of complexity, in large organisations, a lot of time will be spent trying to onboard the employees and change their ways of working. Hence, Excel represents an initial step that can support the management of ALC until the alignment and discipline are in place to leverage the existing tools. Under the assumption that this discipline is in place and all the information is documented in Bizzdesign, it would be preferred to have the ALC scores as part of this tool. By leveraging an in-design approach, IT architects can have more flexibility. PowerBI on the other hand needs to be used for an easier integration with the leadership dashboards.

This means that, for project solution options, Excel is the best option in terms of tooling for quantifying ALC at the moment (for the case study organisation), whereas Bizzdesign is the desired one. Expanding beyond projects and evaluating portfolios and their respective application landscape scores, besides the use of Bizzdesign, PowerBI dashboards can be created to also indicate to the business what is the status of ALC. Furthermore, the ALC evaluation can also be embedded in tools such as ServiceNow which are currently used by the organisation to assist with Strategic Portfolio Management.

To ensure the method is embedded into the company, its practicality must be demonstrated. This can be done by illustrating how to measure the three indicators and clearly stating their sources, as advised by an expert. Subsequently, the architect community should be informed and initiate pilot testing. If refined post-pilot and incorporated into solution designs, the tool is expected to be an effective means of supporting the management complexity.

[7] PowerBI - Interactive data visualisation software part of the Microsoft suite

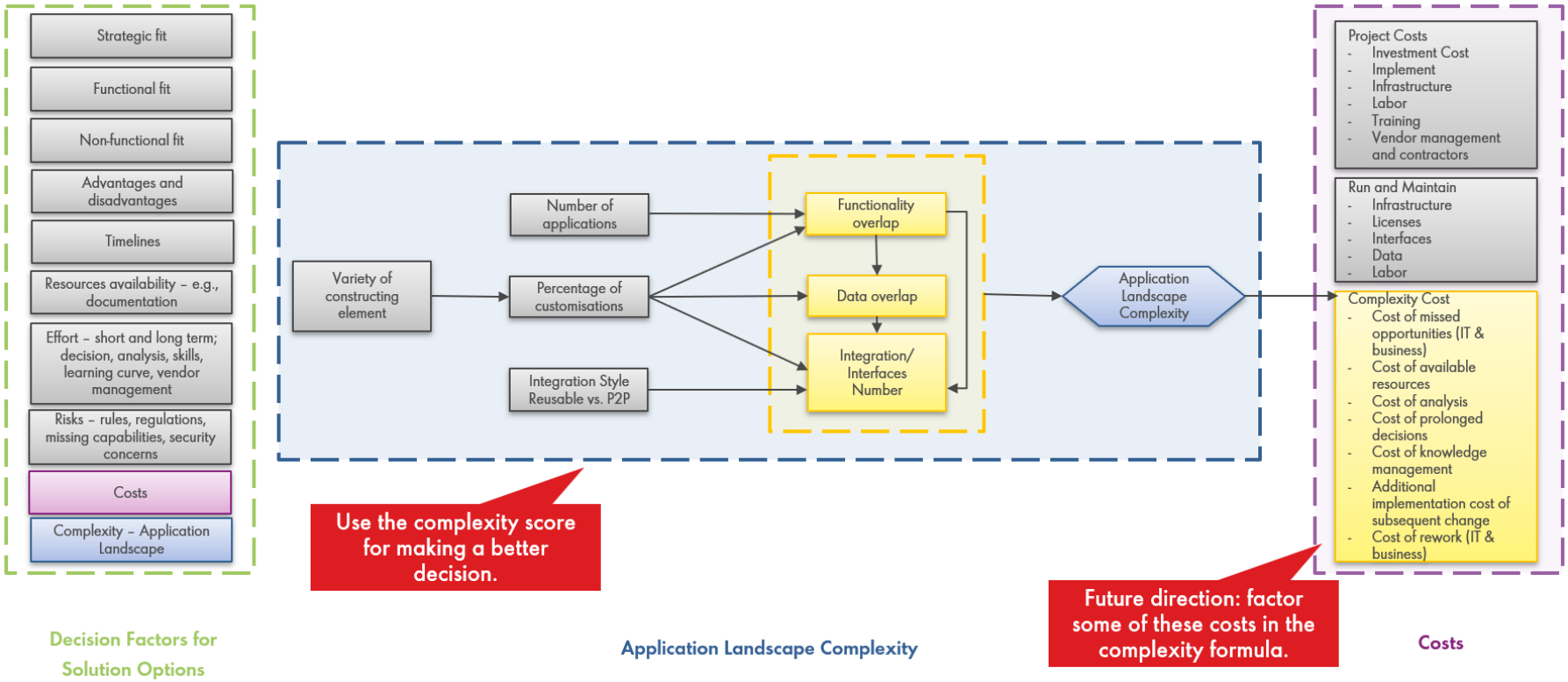


FIGURE 5.10: Solution Options Choice Points

5.5 Final Proposal for Method

By looking at Figure 5.10 and Figure 5.9 a proposal for an improved measurement method of the application landscape complexity is introduced.

5.5.1 Steps

IT architects should first document project solution options and create designs for the current situation and the possible future options. These must at least showcase the application components and their links with application services and data sets. Then for each to-be state, various elements such as alignment of the option with requirements, non-functional fit or the ALC complexity need to be investigated for better decision-making processes.

As the latter is at the heart of this research, the IT architect will be able to generate this score by using the proposed formula. The initial formula for complexity, as seen in equation 4.1, can be calibrated in later stages using the suggestions from Figure 5.9.

5.5.2 Automation for Quantifying Application Landscape Complexity

Creating Views

For Project Current and Future States

As multiple participants in the case study desired to have an automatically generated score for ALC, a possible approach to do this in Bizzdesign is introduced. The GitHub repository[8], mentioned in 4, has been expanded to include the Bizzdesign project (as an xma file), which is detailed below.

First, upon opening Enterprise Studio (Bizzdesign's tool for modelling), an IT architect must create views for the project's current application landscape and for possible future states. In Figure 5.11, one view for an as-is and four to-be situations are depicted. These views are part of a package called "Project Digital Twin" (as the modelled views are based on the example detailed in Section 5.2).

The required elements in these views are application components, data flows, application services realised by application components, and business objects associated with the application components. Without them, the complexity formula can not be calculated.

The four views, in the example model, are simplified versions of Figures 5.1, 5.2, 5.3 and 5.7. This means that Application A will be depicted alongside the data platform, the digital twin, the data object and application service, as seen prior (in the figures mentioned above). However, instead of representing Application B as two components, in the representations in Bizzdesign, they are grouped into one application component called "Application B".

For Transitions

Next, per project, an options view can be designed. In this view, one plateau represents the as-is and the others are possible future states. Each plateau must be linked to its corresponding view. Figure 5.12 illustrates how the as-is plateau is connected with the as-is view.

[8]The researcher can be contacted for collaborator rights/access to the private GitHub or future ideas for tool development. Alternative link: https://gitfront.io/r/evastoica/nX7kKTGXTAYX/ALC-Quantification/

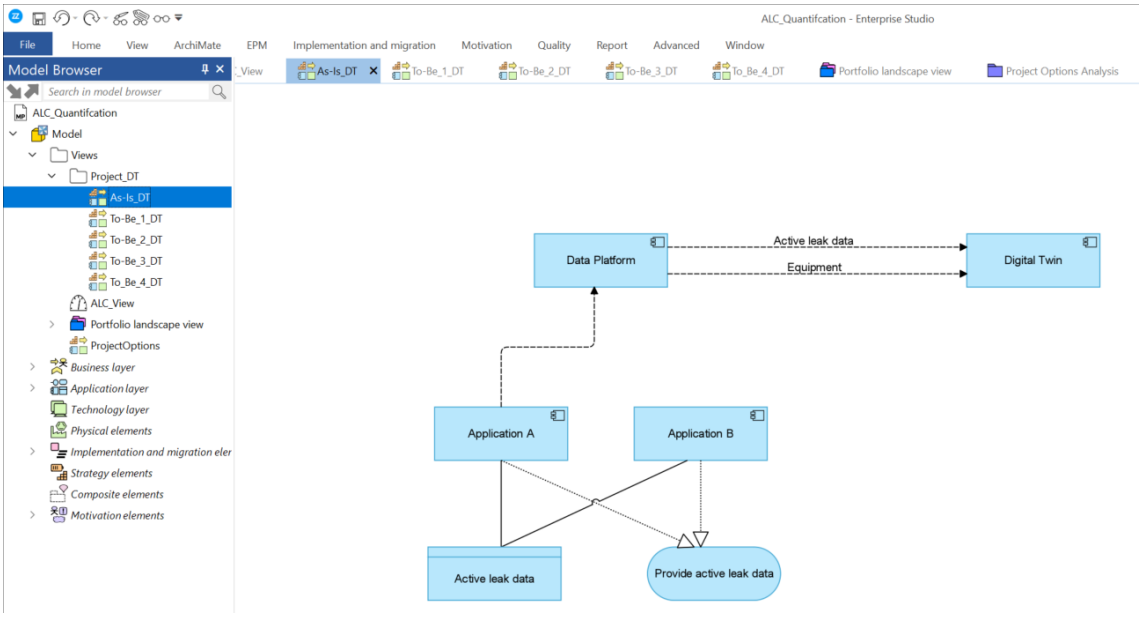


FIGURE 5.11: Views for a Project and its Options

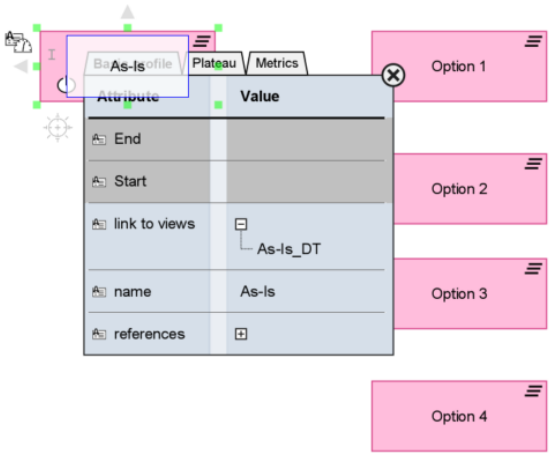


FIGURE 5.12: Linking Plateau with a View

Create Metrics

Next, four types of metrics can be created. The ALC metric is the aggregate metric which quantifies the application landscape complexity. This is based on metrics for counting the number of interfaces, counting the data overlaps and counting the functionality overlaps. By using a metric view, the relationship between the ALC (as a parent) and the other three indicators (as children) must be created. This is showcased in Figure 5.13. Table 5.2 details the terminology mapping from literature to Bizzdesign.

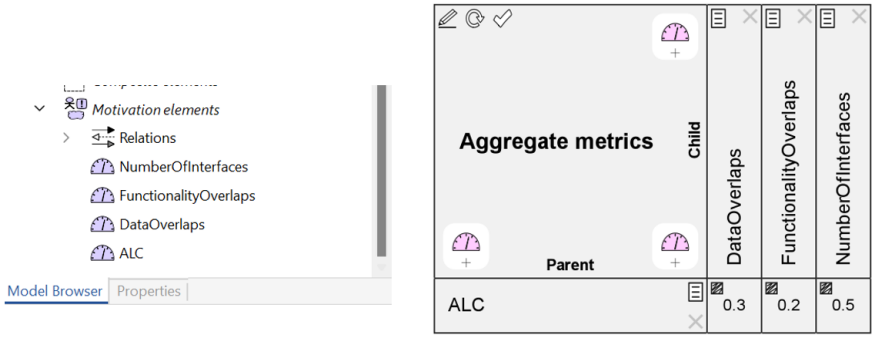


FIGURE 5.13: Creating an Aggregate Metric

Table 5.2: Measurement Terminology in Bizzdesign

Concept	General Terminology	Terminology in Bizzdesign
Complexity	Attribute/Latent variable	Metric (as a parent or aggregate)
Driver of complexity	Indicator/Observable variable/Measurand/Factor	Metric (as a children)
Quantifying the driver	Metric	Measurement (as a script)

Automatic Scoring of Metrics

Next, it is important to mention that ALC is a summation of the values streaming from the children and gives weights to each of them. As seen in equation 4.1, and in Figure 5.13, the weights proposed are 0.2, 0.3. and 0.5. Then, the metrics for quantifying data and functionality overlaps or the number of interfaces will have a script assigned for automatic calculation.

The script represents a small piece of code, written in a programming language to automate tasks. The code used for the scrips of the three indicators can be found in Appendix F. In Figure 5.14 the metric view is presented, showing how ALC is composed of three children, each calculated based on a given script.

The current scripts are tailored to the organisation in the case study, but they can easily be modified according to a certain company's architectural representations in Bizzdesign. Moreover, the current scripts address the issues with granularity in modelling (as different architects might design IT solutions differently). For support on the scripts, the researcher can be contacted. The initial idea of the scripts was to start small and have a basis that can be expanded or improved after subsequent tests.

Connect Metrics to Plateau

To gain insights from these metrics, it is important to go back to the plateau view previously designed. Here, each of the plateaus should be connected with the metric ALC, as seen in Figure 5.15. Then, the architect needs to create a "Portfolio Landscape View" which can be named "Project Options". Then the selected population of the view needs to be the

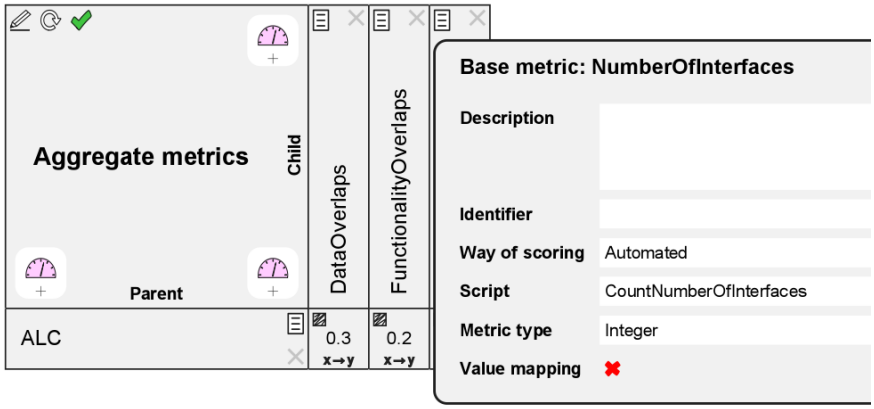


FIGURE 5.14: Scripts for Scoring Metrics

plateau view created in the beginning (see the name of the view in Figure 5.12). Next, the ALC metric must be added to the portfolio view, as depicted in Figure 5.16.

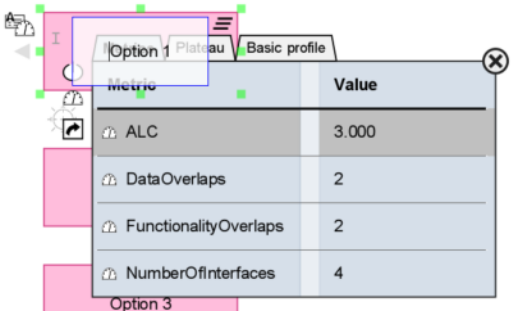


FIGURE 5.15: Metrics per Plateau

Once this is done, by opening the portfolio view and by populating and scoring the portfolio, the score of ALC and its children is calculated, as seen in Figure 5.17. In this option scorecard, the values are shown for concrete examples, the views of the DT project (Figure 5.1, 5.2, 5.3 and 5.7, where Application B is seen as one component). To go even further, a dashboard can be created to showcase these scores (see the spider chart example in Figure 5.18) and further recommendations or comments can be added for each individual option.

In the case of this DT project, Options 1 and 3 had the same score, and the decision was made to choose Option 1. This is due to the approach of transmitting data by creating a reusable interface for Application B and a connection to the data platform, rather than creating a singular P2P connection that can only be accessed by Application B and the DT.

The automation is currently limited to calculating the ALC score. Nonetheless, based on these outcomes, the delta of complexity can be calculated as a difference between plateau scores and added to the "Project Options" portfolio landscape view. Furthermore, in the literature, it was highlighted that for each component part of a view, costs can be depicted as attributes. Hence, if in the as-is and to-be views have several costs assigned by experts, the formula can assist in automatically calculating the costs of the landscape.

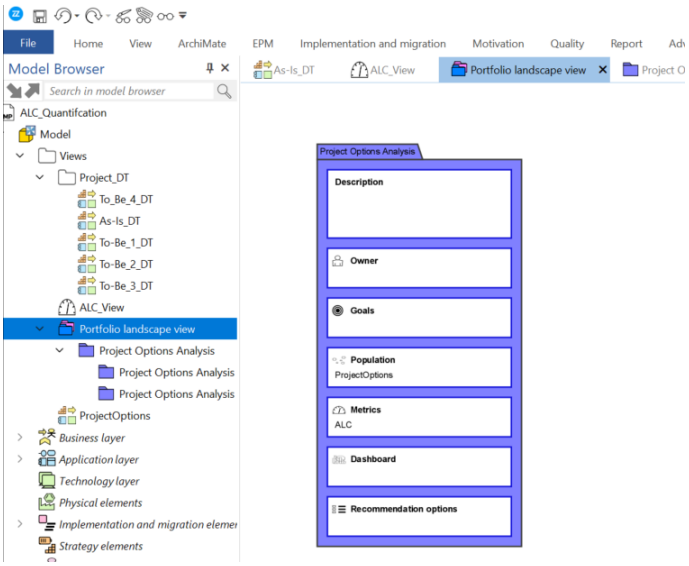
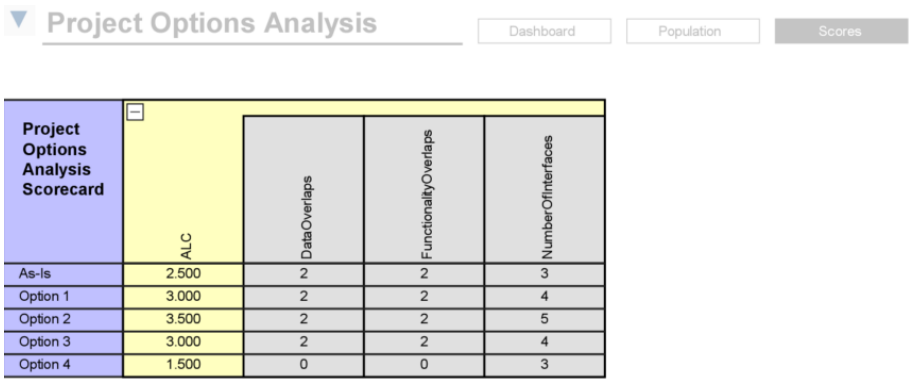


FIGURE 5.16: Portfolio View



Project Options Analysis Scorecard	ALC	DataOverlaps	FunctionalityOverlaps	NumberOfInterfaces
As-Is	2.500	2	2	3
Option 1	3.000	2	2	4
Option 2	3.500	2	2	5
Option 3	3.000	2	2	4
Option 4	1.500	0	0	3

FIGURE 5.17: Options Scorecard

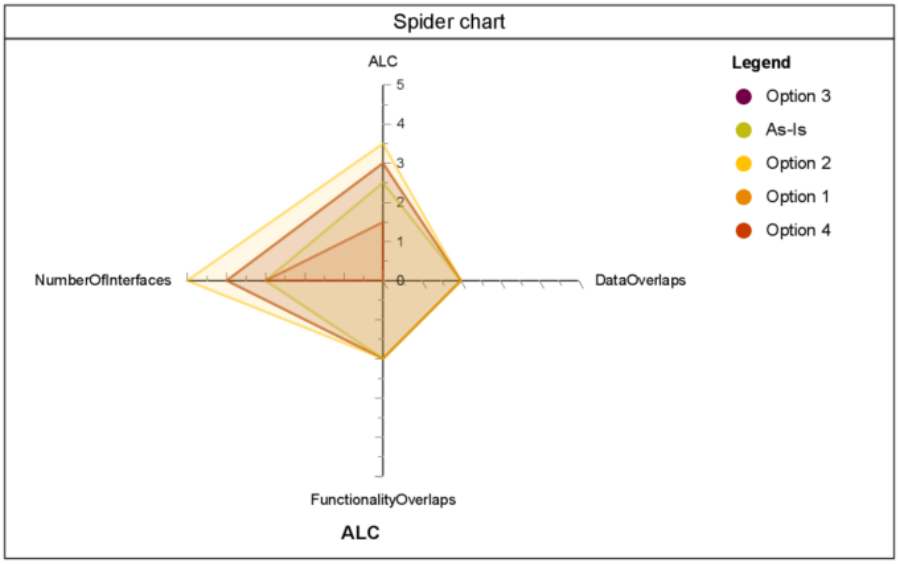


FIGURE 5.18: Spider Chart Scores

5.6 Summary and Takeaways

Chapter 5 describes the outcomes of the validation procedure for the proposed artefact. These emerged after discussions with experts on topics such as meeting requirements, application of the measurement method in different contexts or using similar methods to quantify complexity. The following steps were taken during the validation: start by showing the measurement model and the ALC formula, demonstrate how the method is working based on a case from the enterprise, and have a discussion with the experts about the artefact.

As several key points stood out after the validation procedure, these are enumerated below.

- The most important element in the equation of ALC are the weights assigned to the three factors: the number of interfaces, the data overlaps and the functionality overlaps. These weights must be adjusted to match an organisation's architectural choices.
- The formula for ALC should not factor in relative assessments such as good or bad, adequate or inappropriate complexity. It should be used at an absolute level and offer a score based on which decisions can be made.
- To quantify the cost of complexity, the "hidden costs" should be summed. Furthermore, given the monetary value of the cost of complexity, the ALC score can be calibrated.
- The method is expected to help architects in the management of complexity and in making effective decisions.
- The method proposed can be applied to more than project solutions, for example, it can be used as a portfolio analysis tool. Presently, using an Excel spreadsheet to calculate complexity is the adequate approach, however, in the future using the ALC score as an in-design approach for IT (leveraging Bizzdesign's capabilities) and creating dashboards in PowerBI for the business will bring the most benefits.
- The measurement method and quantifying ALC can be automated in Bizzdesign. This research illustrates how to automate the calculation of the ALC score for as-is and to-be options.

Chapter 6

Discussion

This chapter combines the knowledge gathered from the literature reviews performed during this research (detailed in Chapter 2) with the insights collected from the case study and the validation of the proposed measurement method (as seen in Chapter 3 and 5).

Each section in this chapter details some essential points that underpin the research and their implications for both academia and practice. This means that each section focuses on the key findings of the research, their interpretation and implications. In the end, the most important takeaways from this chapter are listed.

6.1 Complexity Drivers

The first findings addressed are concerned with the possible elements impacting, influencing or causing complexity in the application landscape.

6.1.1 Type of Drivers

One of the most important outcomes of the research, for creating a simpler and more agile IT landscape in an enterprise, is shifting the focus from just rationalisation towards understanding where a larger size, diversity and high number of connections comes from at the application level. Both RQ1 and RQ3 assisted in arriving at this point of discussion.

In both literature and during the case study it became apparent that drivers for complexity are numerous and can roughly fall under two categories: technical/IT drivers and organisational/business drivers. For a better classification, as seen in Figure 3.9, the business drivers can be split into internal and external factors.

The figure also indicates which drivers of IT complexity are mentioned more often in literature. Prior studies in academia, such as [79], have shown the importance and impact of technical drivers on complexity. These could be, for example, an increase in the amount of applications or the diversity of the elements part of a landscape.

Perhaps one of the most striking differences in terms of the technical drivers is that, in practice as opposed to the literature, the number of applications is not considered a direct indicator of application landscape complexity. This study has discovered that the number of applications is relative to other indicators, such as the number of interfaces between applications. This means that having a higher amount of applications may not be an

issue in an IT landscape. Complexity emerges with intricate integrations between these applications or with data issues when exchanging and using information.

With the help of the case study, it was identified the business drivers are the "root cause" that needs fixing. This research demonstrates that drivers such as market maturity, laws and regulations, company culture and politics are not given sufficient attention in the literature, however, they are deemed important in practice. These need to be considered when managing complexity as they can impact the growth of ALC.

Thus, with the simpler driver division in mind (business and IT), changing IT to fix complexity, or addressing the technical drivers of complexity, is perceived as a simpler endeavour as opposed to adjusting the business processes. However, this only helps in managing short-term complexity. To have a simpler overarching IT landscape, it is critical to solve the root causes coming from the business drivers of complexity.

This is expected to support the creation of an IT landscape which will allow agility, effective changes and easier navigation of a dynamic environment. As mentioned by experts during the case study, a 70-30 split should be in place, where 70% of the attention is given, for example, to making a change to business processes and strategies, or to creating handrails and guidelines that need to be followed. The remaining 30% should go towards reducing the number of applications, adjusting the level of customisations or using better integration patterns.

6.1.2 Lack of Documentation

In this study, another finding that should be discussed is the impact of documentation on application landscape complexity. In Figure 3.5, the researcher also included the lack of documentation as a potential indicator of complexity. This was seen as the absence of documentation about the application landscape, such as missing information about a system. In the literature, the link between complexity and documentation is not specifically mentioned. Hence, when answering RQ3, with the help of the case study, more clarity on the link and relationship between documentation and application landscape complexity has been created.

During the survey, the outcomes showed that lack of documentation was ranked as one of the least important indicators for application landscape complexity. During the interviews, this was further investigated. It became apparent that lack of documentation is perceived as an enabler rather than a direct driver of complexity. This means that it only allows the complexity of the application landscape to stay "hidden" or difficult to understand.

This result shows that issues with documentation, such as the lack of it, are elements that influence how an architect perceives and comprehends the application landscape. Other problems such as how much the documentation covers (e.g., information about the current application landscape such as specifications of IT components which can be gathered from platforms/EA representations in the company), its quality or its age (e.g., the latest update) are also important and make the management of complexity an easier or harder task for architects.

6.2 Formula for ALC

Next, the discussion on the proposed formula for quantifying ALC is introduced.

6.2.1 Indicator's Definition

An interesting outcome is related to the level of subjectivity and personal interpretation of what the complexity of the application landscape means. To achieve a common understanding, at a company level, and make more informed decisions based on ALC, an enterprise must create clear definitions of complexity indicators. These need to be shared, adopted, and used in the whole organisation. This point is touched upon in both the literature and the case study.

As observed during the case study (see the complexity capability maturity model in Figure 3.4), individuals refer to various elements when explaining what application landscape complexity stands for. For some, this only means a high number of applications in their area of work, whilst for others complexity goes beyond the size of the landscape and is also related to interconnections between applications, percentage of customisations etc. With clear definitions of complexity and its indicators, an organisation can start improving their complexity management capabilities.

A useful tool that can help is the complexity capability maturity model described in the literature [3]. Based on this model, an organisation can move step by step to increase its capabilities in managing application landscape complexity.

Furthermore, once individuals have an aligned view of what each indicator of ALC means, measuring these becomes an easier task. On the basis of such definitions, a better overview of what should be in place when modelling architectures can also be created. These points are supported by the discussions held during the case study. These illustrated the fact that more discipline is needed for EA modelling, as architects have the freedom to depict a landscape according to their views.

For example, quantifying the application functionality overlap, without a clear definition, can mean for some architects, counting the number of applications serving the same business function, whilst for others it can mean the count of the delivered services or maybe even the realisation of capabilities. Hence, a definition sheds light on what exactly needs to be quantified and what must be included in designs (e.g., depict services, use capabilities etc.).

6.2.2 Indicator's Weight

Once having a set of definitions for indicators of application landscape complexity, organisations can start quantifying each of them. The prevalent literature and the case study show that the application functionality overlap, data overlaps and the number of interfaces are some of the most important indicators of complexity. From the literature, as shown by Schneider et al. [82], it also became evident that numerous organisations have are collecting data to quantify these three indicators. Considering the indicator importance, the proposal for the formula of ALC can be generalised, and it is expected to help various enterprises.

Nonetheless, one aspect of the formula that must be adapted is the weight given per indicator. For example, in large companies such as Netflix or Uber, which decided to implement microservices [160],[161], having a higher number of interfaces should not weigh the most. Having a higher number of interfaces is not considered more complex. In this case the applications delivering the same services must weigh more.

Based on the company's choices for its IT architecture, the weights can be adjusted for

various use cases. These cases can be designed considering how the business operates the IT or how the teams within an enterprise are structured. For example, a use case can be created by looking at the type of patterns used in the company, and the architectural style decisions present in an enterprise (e.g., use of data mesh architecture or microservices). Moreover, the choice in terms of using domain-driven design[9] or traditional approaches to enterprise modelling can be analysed [162].

For the architectural style, the weights must be adapted as explained previously, in the microservice example. Also, in the case of choosing a data mesh as a data management approach or using common data models in an organisation, a higher number of interfaces will not weigh as much in the ALC formula, whereas the significance of data or functionality overlaps will carry a greater score. For integrations, different weights can be given considering elements such as application coupling, integration simplicity, technology, data format, timeliness or synchronicity as mentioned in [141]. The organisation at which the case study was conducted also had an integration guideline book in which handrails were provided on how to enable an appropriate collaboration of applications.

6.2.3 Solution Options

During the case study, the proposed method for quantifying application landscape complexity has been perceived as a support for solution options analysis. This method can help with the guidance of IT architects towards effective decisions. In both the literature and the case study, complexity is difficult to quantify and manage. Thus, with the help of the formula which can generate a score per solution, an architect can understand the level of ALC.

The score is expected to guide architects when deciding on what changes to implement in the application landscape. However, this score is not the only decision factor. As illustrated in Figure 5.10, several other choice points influence a decision. These can range from functional or non-functional fit to risks.

Hence, the proposed artefact can become a part of a "Pokemon Card" (as mentioned by one of the experts during the validation of the artefact). Pokemon cards contain information about a creature type, ability or any further characteristics [163]. As such, the respondent envisioned a similar card containing all the information of a to-be option.

In this card, ALC is a quantitative factor guiding future choices in projects. Other qualitative elements, such as strategic or functional fit are elements in this card. Each option can have such a "Pokemon Card" based on which the IT architect can choose a to-be direction for a simpler and smarter IT landscape. Figure 6.1 has been designed as an example of how such a card can look like. Based on the combination of the results of qualitative and quantitative factors in a "Pokemon Card", the architect can make an informed decision towards the future.

6.2.4 Automation

During the case study, when answering RQ 5, it became apparent that the management and quantification of complexity is an easier endeavour for organisations which have architects following certain guidelines. These types of guidelines assist in aligning the views of

[9]Domain-drive design - Approach for building complex architectures around business concerns, which got more attention because of its adoption by microservices style

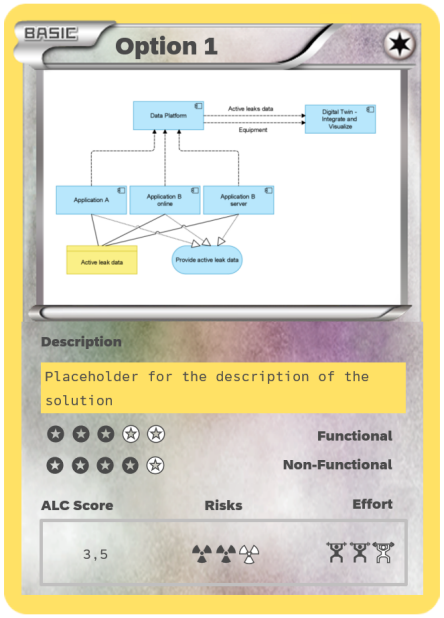


FIGURE 6.1: Pokemon Card High-Level Design

the organisation on the application landscape as much as possible (e.g., how to integrate applications, what needs to be in place when modelling a change of the IT etc.).

Subjectivity plays an important role in how the IT architecture is represented throughout the organisation, as architects can make choices freely. This can lead to departments using different IT solutions that support a certain business process. By comparing various parts of large organisations, without using certain architectural guidelines, discrepancies in application landscapes can be observed. For example, some parts of the organisation can use middleware, manage their data in an adequate manner etc., whilst others can create tightly coupled solutions by having applications deployed on the same server and using code modifications to create P2P connections. This can be illustrated by considering the integration guideline book of the company at which the case study was conducted. It is expected that the architects will leverage this integration book when designing solutions. However, it can be the case that these types of guidelines are not used if the employees in a large company are not accustomed to or are not aware of them, amongst other factors.

To exemplify the importance of guidelines further, the case of automatically generating a score for the ALC, as discussed in 5.5.2 can be broken down. First, without handrails, this automation procedure can not exist. If the architects only model a target state and do not consider several options, there will be no added benefit of calculating complexity. The method proposed in this thesis should be part of a solution option tool for effective decisions. Moreover, if the solutions do not include certain elements such as application components, application services, data flows to depict interfaces, or data objects (in the case of the company at which the case study was conducted), the automatically generated value of ALC will be inconsistent when used in several parts of the organisation. This means automation can be in place if a set of required elements are present in an as-is or to-be architectures.

Hence, to automate the procedure for calculating ALC, handrails and documentation are necessary, such that architects will know what is expected of them and how they can reach

a better decision-making process. These guidelines ensure that expected benefits, in the case of this research support for the management of complexity, can be achieved in an enterprise.

6.3 Impacts of Complexity

Next, the impacts of application landscape complexity on the enterprise are discussed.

6.3.1 "Hidden Costs" of Complexity

Moving to the economic pillar of an organisation, when discussing about an IT landscape and its complexity, the majority associates it with a higher total cost of ownership. An emphasis is put on how a complex landscape leads to more run and maintain costs or project costs. In the literature the same costs, for example, component, licenses, integration, and migration, were the most highlighted ones.

However, during the research, several other costs driven by a complex application landscape (as a part of the IT landscape) were identified. If the landscape is complex and a change must be made, the time spent in making decisions might be longer due to trying to understand where to start. Moreover, certain opportunities can be missed as a complex landscape might not permit being agile.

Therefore, architects and managers need to acknowledge these "hidden costs", linked with the intricacies of change processes, which can range from costs of prolonged decisions to costs of rework, as seen in Figure 4.9 and in Table 4.4. As they are harder to quantify as opposed to the project or run and maintain costs, more discipline is required to generate a monetary value for the cost of complexity.

6.3.2 Management of Complexity

The overarching result of the thesis should serve as a first step towards a better management of complexity and a more effective decision-making process for a simpler IT. At the moment, the formula gives an absolute value to application landscape complexity, but this should also be interpreted.

Complexity, as mentioned throughout the thesis, can be seen as both a positive and a negative element in an organisation. During the case study, it was observed that the architects should balance opportunities and requirements alongside dogma and pragmatism to create an adequate level of complexity. In literature, this is named "requisite complexity" [66], [18].

To reach this point in which somebody can categorise complexity as "good or bad" based on the outcome of the formula, multiple cases need to be analysed and connections with the performed changes must be created. For example, if the ALC score is applied to several project options and the architects explain what changes they made per each option, a better understanding is created regarding the meaning of a low or high score of complexity.

6.4 Summary and Key Takeaways

Chapter 6 discusses several results of the research and their implications prior to reaching the conclusions. The list below summarises this chapter and illustrates the essence of the discussion points.

- Performing tasks such as rationalising the IT landscape (e.g., reducing the number of applications) is expected to influence the overall complexity for a short time. For the management of complexity, it is important to address the business drivers, as those are the ones that, when acknowledged and understood, can create a simpler, more agile IT landscape.
- The number of applications may not be a direct indicator of complexity. This is relative to aspects such as the interconnections between applications.
- Issues with documentation such as the absence of it, its inadequate quality, insufficient coverage or infrequent updates are hiding how complex an application landscape is. This leads to a more difficult process of managing complexity.
- To quantify ALC, a clear definition of what it means and how it can be broken down into smaller indicators is needed. Based on these indicators, an architect knows what needs to be measured and, thus what is necessary to model in a landscape.
- The weights indicated in the initial formula for ALC need to be adjusted based on different use cases which can be observed at an organisation level. These use cases are concerned with the decisions taken by a company to structure their IT landscape (e.g., decomposition patterns – use of microservices etc.).
- The ALC score can be part of a solution "Pokemon card", alongside elements such as strategic or functional fit. Based on this card assigned to a future state (e.g., a view depicting the to-be of the application landscape), the architects can make better-informed decisions.
- To automate ALC, guidelines and handrails need to be in place and must be used to indicate how to model solution options in a tool such as Bizzdesign. These guidelines make sure that everything is modelled similarly and that quantification of ALC can be performed based on existing models.
- There are several costs that should be acknowledged when calculating the costs of complexity. These range from the cost of missed opportunities to the costs of rework or the costs of prolonged decision processes.
- The calculated value of ALC should be interpreted (not seen as an absolute value) only after multiple tests. During these, it should be identified what types of changes in a landscape are associated with a higher or lower ALC score.

Chapter 7

Conclusion

The last chapter of the thesis concludes the research by providing the lessons learned by answering to the RQs, in section 7.1. Upon providing the answers, the contributions to both research and practice are highlighted in Section 7.2. The remainder of this chapter details the limitations of the study and the future avenues of exploration 7.3 and 7.4.

7.1 Lessons Learned

This thesis has set out to design a method for quantifying application landscape complexity and understanding its cost implications. This should be applied in a practical setting and assist IT architects in making effective decisions for keeping the landscape agile for future changes. Due to the dynamics and growth of IT developments, enterprises must have the ability to quickly adapt to changes. Research shows that managing the complexity of an IT landscape is a difficult endeavour for organisations and that there is a lack of tools that support this.

Therefore, the thesis aimed at responding to the following overarching research question: *"How can a practical measurement method be designed to quantify the enterprise architecture complexity and its cost implications?"*. Several sub-research questions have been formulated to help capture all the needed information for crafting a comprehensive answer to the main RQ.

7.1.1 Complexity Metrics and Costs

The first sub-research question *"Which complexity metrics can be used to measure change in an IT landscape?"* explored existing indicators of complexity. Next, the second sub-RQ investigated *"What types of costs are associated with the IT landscape?"*.

These first two questions have been addressed in Chapter 2. Starting with complexity drivers and metrics (sub-RQ 1), in literature, IT complexity is seen through a myriad of lenses. As no study offers a comprehensive overview of elements that drive this type of complexity or how it can be quantified, several works have been combined to answer the question. It became apparent that potential drivers for IT landscape complexity are, for example, size, diversity, and heterogeneity. For metrics of complexity, aspects such as counting elements or entropy have been identified.

The second sub-RQ explored the types of costs associated with the IT landscapes. When

working with IT landscapes, the costs that are accounted for are the "most obvious ones", primarily falling under run and maintain category. It is rarely the case that costs connected to change processes or complexity are acknowledged or are guiding decisions. Moreover, costs are determined independently of complexity and the assumption is that if the number of links or interactions in an IT landscape increases there will be a growth observed in costs. Another unanticipated discovery when performing the literature review for sub-RQ 2, was that there is a lack of techniques that cover quantitative cost analysis in architectural models.

7.1.2 Measuring ALC

The third sub-RQ *"What are suitable complexity metrics that can be used in measuring the delta of the change in the application landscape?"* assessed possible combinations of metrics for application landscape complexity. It was uncovered, during the case study (Chapter 3) that there are three metrics, which are considered adequate for displaying the level of complexity of an application landscape.

These are counting the number of applications which have functionality overlaps (how many applications are realising the same service), quantifying the applications' data overlaps (e.g., by counting how many applications are there per a data set/entity/object) and lastly, evaluating the total number of interfaces. By aggregating the metrics of these three indicators (application functionality overlap, data overlap and interfaces), an ALC score for a design can be generated. Afterwards, the difficulty in the move from an as-is to a particular to-be is calculated by subtracting the complexity of the initial state from the one of the future state.

The company involved in the case study showed interest in quantifying complexity with the use of this formula, recognising its potential for broader applications beyond projects.

7.1.3 Costs of Complexity

The fourth sub-RQ, examined the costs driven by application landscape complexity *"What costs can be factored in a formula when application landscape complexity hinders the agility to adapt to a change?"*. During the case study (Chapter 3), it became apparent that due to the complexity associated with the application landscape an array of "hidden costs" should be acknowledged by architects. When making choices, besides looking at the total cost of ownership, several costs such as the one of missed opportunities, prolonged decisions or rework must be factored in. The proposal is to use the hidden costs in a sum which indicates the cost of complexity.

7.1.4 Practical Use of Arterfact

The last sub-RQ explored *"To what extent can complexity metrics be used in analysing the enterprise architecture in the context of a large enterprise?"*. The findings, formed during the validation process in Chapter 5, indicate that the metrics can be used effectively only if a company has comprehensive documentation and relevant data on the complexity drivers and indicators.

Furthermore, IT architects must adhere to guidelines when designing IT landscapes. By looking at the scope of the research, the data needed for the three indicators should already be visible in an organisation. If not, the employees in the organisation must first establish

a common understanding of application landscape complexity. Once defined, the method to quantify complexity can be integrated into the current ways of working.

Guideline creation is a recommendation for companies dealing with complex IT landscape. One should start by defining what complexity means, as shown in the thesis: ALC means the number of interfaces and the applications with data and functionality overlaps. Next, the guidelines for applying the method should be documented. In the case of this research, the steps proposed in Section 5.5.2, detail how solution options should be designed in a modelling language and tool to allow ALC quantification. Following this, the method should be tested for smaller scopes such that its benefits can be shown. Once the definition, the documentation are adequate and results are visible for smaller scopes within the company, promoting the measurement methodology to ensure is use is next. This way, all architects will know the required design steps, necessary elements, and the benefits of ALC quantification.

7.1.5 ALC Measurement Method

With this array of answers, the response to the main RQ lies in the design of the artefact proposed in the thesis. The measurement method introduced in this study quantifies a section of enterprise architecture complexity by looking at the application layer. The current way to quantify ALC in a practical setting is to consider designs of solution options in projects. Per the current state and the different future options, a complexity score is calculated by architects using the formula and approach proposed in the thesis. Given the complexity score, one can also investigate what is the monetary value of complexity if the "hidden costs" driven by complexity are documented.

Hence, when designing a measurement method for complexity that can be applied in a practical setting it is important to scope the landscape down, clearly define drivers, indicators and metrics of complexity, embed the approach into the current ways of working in a company and improve the approach incrementally.

Serving as a starting point, the artefact can be enhanced or developed further by practitioners. More tests and expert interviews must be conducted before the large-scale implementation of the method. In organisations or business areas having a certain level of enterprise architecture discipline, architects or managers can begin using the proposed artefact in various cases to evaluate its outcomes.

7.2 Contributions

By creating such a comprehensive list of answers, the thesis contributes with valuable insights to both academia and practice and is expected to be a foundation for future works. These contributions are detailed in the subsequent sections.

7.2.1 Literature Study and Conceptual Model

In the beginning, the mixed method literature review pieced together various dispersed streams of knowledge on the topic of IT landscape complexity. As numerous perspectives exist on what the IT landscape means, this thesis contributes with a conceptual model, as seen in Figure 1.2. By creating such a model, the views are linked together and the first steps towards an alignment of perspectives on IT landscapes is achieved.

7.2.2 Models of Drivers and Metrics of Complexity

General

Furthermore, by performing both a literature review and a case study, an enhanced overview of complexity drivers and indicators was developed, as seen in Figure 3.9. This brings additional insights to light as the business drivers which influence complexity and their importance are not thoroughly emphasised in the literature. The thesis uncovered that addressing business-related issues is crucial for the long-term agility of the IT landscape. The landscape is expected to remain simple and agile for a longer period if the root causes streaming from the business side are solved, as opposed to only reducing the number of application components or improving integration patterns.

ArchiMate

The research also contributes with an ArchiMate model for illustrating IT landscape complexity drivers and metrics. This can be seen in Figure 4.1.

7.2.3 List of Costs of Complexity

Additionally, insights into various IT landscape costs were synthesised. From the literature, the approaches and connotations of "the cost of complexity" are illustrated. Furthermore, a proposal is made of the elements described as "hidden costs" that are expected to emerge due to a complex application landscape. Table 4.4 offers an overview of costs emerging due to a complex IT landscape, which need to be more actively accounted for/monitored in large organisations.

7.2.4 ALC Structural Model

The research also provides the first measurement model, as seen in Figure 4.2, combining indicators of application landscape complexity, detailing the influences among factors and complexity. The model identifies application functionality overlaps, data overlaps and the number of interfaces as relevant and sufficient indicators to quantify application landscape complexity.

7.2.5 Architectural Patterns

Based on these three indicators, the research also illustrates three patterns for showing what a complex versus simple representation is. These can be found in Figure 4.3, 4.4, 4.5. Furthermore, an additional 4th view comparing a simple versus a complex application landscape component is proposed for differentiating the integration styles used by a company, as seen in Figure 5.8.

7.2.6 Measurement Model and Formula

Based on this ALC structural model, a method (which can be implemented manually or automatically) for selecting options by quantifying complexity is proposed. Organisations can reuse this method and adjust it to fit their enterprise architecture landscapes.

The thesis also contributes to the existing body of knowledge by introducing the formula to quantify ALC as a part of the measurement method. Two approaches are proposed by the thesis to apply the method in a practical setting. The first is the use of Excel, whereas the second is the method applied in an EA modelling tool (Bizzdesign). A comprehensive

step-by-step guide of how to use Bizzdeign for quantifying ALC is introduced in the thesis, as seen in Chapter 5.5.2.

For the enterprise architecture practice, this research thus offers a method that can assist architects in quantifying ALC and in choosing future options. As complexity is a rather invisible aspect in an organisation and hardly any complexity management tools exist, architects or managers can use the artefact proposed to make complexity tangible.

7.2.7 Discussion Points

Lastly, the discussion points mentioned in Chapter 6 illustrate several key results of the thesis and their implications. By having the interpretation of the results, a deeper understanding of the topics investigated throughout the thesis is achieved. Furthermore, it showcases how the existing literature intertwines with the case study outcomes, adding depth to the research.

7.3 Limitations

The limitations of this research need to be addressed, hence this section discusses several areas which could have been improved.

7.3.1 Qualitative Study

The first limitation is surfacing from the choice of conducting the thesis as a qualitative study. The mixed-method literature review and the case study are done through the prism of only one researcher, due to the nature of the thesis. Having this singular, subjective, perspective on gathered results can be said to include a level of bias or maybe omit insights. To ameliorate these concerns, both the literature reviews and the case studies were performed according to rigorous guidelines that are expected to help with the reduction of bias and to create study applicability.

7.3.2 Participants

Another limitation experienced is concerned with the number and type of participants involved in the survey, interviews and the validation procedure. 32 respondents actively engaged in the case study. However, they were all working at the same company, which might pose a challenge in capturing diverse perspectives on the topic of application landscape complexity. To try and address this limitation, a choice of talking with experts with various backgrounds, active years and roles in the company was made. Nonetheless, the artefact proposed could have been enhanced by reaching out to people working in other sectors to ensure the completeness of the model and to have a stronger claim that the three indicators for application landscape complexity are sufficient.

7.3.3 Indicators' Links

Related to the artefact proposed, the measurement model currently depicts the most relevant relationships between indicators of complexity based on the outcomes of the case study. As mentioned earlier, this is done by only one researcher and validated with a smaller group of people, hence it can be the case that certain relationships are missing. During the case study, it became apparent that all the indicators are intertwined to a

certain extent, but for simplicity and practicality, only the most discussed relationships are depicted. Hence, the model is expected to be in close proximity with reality.

7.3.4 Given Examples

Streaming from the testing of the proposed formula and the tool, another limitation is concerned with the number of projects and examples that were provided for analysis purposes. Only one example was explained in more detail by an architect and the relevant details were filled in the Excel tool. At the company at which the case study was conducted, EA modelling languages were not commonly used for creating solution options. The project option shared had to be transposed in ArchiMate by the researcher. As EA designs and models are more simplified than reality, it would have been important to look at other documented cases and apply the method for quantifying complexity.

7.3.5 Validation

Lastly, validation inputs came from an expert group (the advisory board) and only some of the interviewees. Due to the limited time frame of the thesis, the method validation could not have been discussed one-on-one with all the interviewees. To address this concern, a presentation, illustrating the relevant findings of the thesis, was sent to all the participants in the study. Upon receiving the presentation, they were asked to respond to a few questions. Thus, validating the method with a larger pool of experts might have enriched the data and created a more comprehensive understanding of the subject matter.

7.4 Future Avenues of Exploration

As there is abundant room for future progress in managing enterprise architecture complexity, based on the thesis findings, their broader ramifications and interesting avenues of exploration need to be presented. These represent collections of sparked ideas during the literature review and the case study.

7.4.1 Ontologies

Starting with future research, for the management of complexity having a clear understanding of this term's meaning is essential. Hence, ontologies can define a common language and align views on an organisation's IT landscape complexity. These are a means to provide a complete and consistent set of terms that define a problem domain, illustrating their relationships and properties [164], [165]. Furthermore, with the help of ontologies, knowledge representation will facilitate improved governance and decision-making processes, alongside collaboration and scalability [165].

Ontologies can enhance the semantic expressiveness of the IT landscape components conceptual model introduced in Chapter 1. By analysing application layer components in this model, an ontology for ALC including indicators and metrics can be explored. This can be extended using Figures 3.9 and 3.10 as starting points for an overarching IT landscape complexity ontology. The work of Azevedo et al. [166] proposing an ontology for ArchiMate can support this. The ALC ontology will be a domain-specific ontology part of the top-level ontology of IT landscape complexity [167], which can also include the type of costs associated with a complex IT landscape.

7.4.2 Measurement Model

It would also be interesting to validate the relationships between the indicators of complexity from the measurement model. Leveraging this type of validation could make it easier to select the appropriate indicators for calculating ALC and showcase if any missing relationships are not part of the model. Furthermore, each indicator should be individually considered and its relationship with complexity should be detailed. For example, how would one of the indicators influence the growth of complexity, is it in a linear, exponential, quadratic etc. manner?

7.4.3 Complexity's "Hidden Costs"

Another important issue for future research is identifying how the proposed "hidden costs" of complexity can be measured. It would be interesting to explore how from a given ALC score, a monetary value of the cost of complexity can emerge. There is no formula that can translate a complexity score into a monetary value. Something along these lines would be valuable for enterprises as it can show how complex a state is and what are the implications on costs. Having such an approach will give an organisation a better sense of direction and will showcase what a high or low complexity score means as a monetary value.

7.4.4 Business Complexity

To develop a full understanding of IT landscape complexity, which spans beyond the application landscape, the mechanisms behind business drivers (e.g., business processes, structures etc.) need to be untangled as they are primarily influencing this type of complexity. For a long-term impact, more agility and less costs, tackling and improving the root causes streaming from organisational factors can reduce the IT complexity. A further study on the connection between ALC and business complexity is needed to confirm these claims.

7.4.5 Calibration of Weights

Looking at the artefact proposed, further investigation is desired to determine adequate weights for each of the three indicators: the application functionality and data overlaps and the number of interfaces. Based on discussions with other companies, use cases for calibrating the weights should be created. Having a holistic view of how several organisations are structured, how they are conducting their operations or what architectural styles they are using for their IT landscapes can help a researcher in crafting these use cases. To enhance the selection of appropriate weights, academic insights can also be collected based on discussions with domain experts.

7.4.6 Real-life Applications

Given the method proposed, this must be applied in real-life contexts. To apply the method, a company must define the meaning of ALC complexity, create guidelines for modelling representations and show its results in real-life situations (as described in 7.1.4).

According to DSRM's design cycle [67], this real-life testing represents an important step. The applicability of the artefact in several real-life cases creates a better understanding of how to form a judgement regarding the ALC score or what elements are commonly associated with a higher or a lower score. By employing this real-life setting, it can also be ascertained whether a change in the complexity of a project impacts the portfolio or

organisation in a particular manner. By adding this observation layer to the simple score of ALC, businesses can also start using the method for mastering complexity.

7.4.7 Scope Expanison

Another direction for future work is the expansion of the methods' scope to a portfolio level. The current artefact was designed as a project solution options tool, which assists architects in making decisions and managing complexity. In the future, the formula proposed for quantifying complexity can be adjusted such that it can be applied to portfolios. If one portfolio is assigned a complexity score, comparing and contrasting best practices should be possible. Architects can learn from each other when a portfolio in a similar line of business has a lower complexity score.

If several other companies can be contacted, the possibility of using the ACL score as a benchmark should be mentioned as an expansion point. If the weights of the indicators are aligned in all the enterprises, such a benchmark can promote a discussion on how each application landscape is designed, what it is comprised of, and what some best practices that could be employed by all the other enterprises in similar contexts are (e.g., understand why a ALC score is lower and what can be done about it).

7.4.8 Artificial Intelligence

Lastly, a promising future direction is integrating artificial intelligence (AI) with ALC scores for solution option choices. If one EA design is assigned an ALC score, AI can suggest ways to reduce complexity, given a training set of past cases associating scores with detailed change descriptions. This requires a lot of discipline and information. AI can aid architects during modelling processes. Similar to computer-aided design, the AI can proactively suggest how to reduce complexity, explain why designing an element in a particular manner might not be adequate and propose suggestions. For example, the AI can mention "You have done a good job in designing this solution, but here are some thoughts on how you might want to improve to reduce complexity, allow agility for the change process and reduce costs". Even more, AI might suggest a to-be option from a current as-is state, given a certain list of requirements.

Bibliography

[1] T. O. Group, "ArchiMate® 3.1 Specification," 2019. [Online]. Available: https://pubs.opengroup.org/architecture/archimate31-doc/toc.html

[2] A. Schneider, M. Zec, and F. Matthes, "Adopting Notions of Complexity for Enterprise Architecture Management," in *20th Americas Conference on Information Systems, AMCIS 2014*, Aug. 2014.

[3] A. Kluth, J. Jäger, A. Schatz, and T. Bauernhansl, "Evaluation of Complexity Management Systems – Systematical and Maturity-based Approach," *Procedia CIRP*, vol. 17, pp. 224–229, 2014. doi: 10.1016/j.procir.2014.01.083. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2212827114003357

[4] M. Lankhorst, *Enterprise Architecture at Work*, ser. The Enterprise Engineering Series. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. ISBN 9783642296505 9783642296512. [Online]. Available: http://link.springer.com/10.1007/978-3-642-29651-2

[5] S. A. Sheard and A. Mostashari, "7.3.1 A Complexity Typology for Systems Engineering," *INCOSE International Symposium*, vol. 20, no. 1, pp. 933–945, Jul. 2010. doi: 10.1002/j.2334-5837.2010.tb01115.x. [Online]. Available: https://incose.onlinelibrary.wiley.com/doi/10.1002/j.2334-5837.2010.tb01115.x

[6] G. Adomavicius, J. C. Bockstedt, A. Gupta, and R. J. Kauffman, "Making Sense of Technology Trends in the Information Technology Landscape: A Design Science Approach," *MIS Q.*, vol. 32, pp. 779–809, 2008. [Online]. Available: https://api.semanticscholar.org/CorpusID:10592597

[7] S. Bente, U. Bombosch, and S. Langade, "Collaborative Enterprise Architecture: Enriching EA with Lean, Agile, and Enterprise 2.0 practices," 2012. [Online]. Available: https://api.semanticscholar.org/CorpusID:107238979

[8] J. Ross, P. Weill, and D. Robertson, *Enterprise Architecture as Strategy — Creating a Foundation for Business Execution*, May 2006. ISBN 1-59139-839-8

[9] A. Schneider, A. Gschwendtner, and F. Matthes, "IT Architecture Standardization Survey," Technical University of Munchen, Technical Report, Oct. 2015.

[10] I. Hanschke, "IT Landscape Management," in *Strategic IT Management: A Toolkit for Enterprise Architecture Management*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 105–217. ISBN 978-3-642-05034-3. [Online]. Available: https://doi.org/10.1007/978-3-642-05034-3_4

[11] S. Laan, *IT Infrastructure Architecture - Infrastructure Building Blocks and Concepts*, 2nd ed. Lulu.com, 2013. ISBN 1-291-25079-4

[12] K. Beetz and L. Kolbe, "Towards Managing IT Complexity: An IT Governance Framework to Measure Business-IT Responsibility Sharing and Structural IT Organization." in *17th Americas Conference on Information Systems 2011, AMCIS 2011*, vol. 1, Jan. 2011.

[13] T. Widjaja, J. Kaiser, D. Tepel, and P. Buxmann, "Heterogeneity in IT Landscapes and Monopoly Power of Firms: A Model to Quantify Heterogeneity," in *International Conference on Information Systems, ICIS 2012*, vol. 1, Jan. 2012.

[14] J. Ross, C. Beath, and M. Mocker, "Creating digital offerings customers will buy : find the sweet spot between what technologies can deliver and what your customers need," Jan. 2019.

[15] K. Wehling, D. Wille, C. Seidl, and I. Schaefer, "Decision Support for Reducing Unnecessary IT Complexity of Application Architectures," in *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*, 2017. doi: 10.1109/ICSAW.2017.47 pp. 161–168.

[16] P. Kruchten, R. L. Nord, and I. Ozkaya, "Technical Debt: From Metaphor to Theory and Practice," *IEEE Softw.*, vol. 29, no. 6, pp. 18–21, Nov. 2012. doi: 10.1109/MS.2012.167 Place: Washington, DC, USA Publisher: IEEE Computer Society Press. [Online]. Available: https://doi.org/10.1109/MS.2012.167

[17] R. Verdecchia, P. Kruchten, P. Lago, and I. Malavolta, "Building and evaluating a theory of architectural technical debt in software-intensive systems," *Journal of Systems and Software*, vol. 176, p. 110925, Jun. 2021. doi: 10.1016/j.jss.2021.110925. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121221000224

[18] E. Onderdelinden, B. v. d. Hooff, and M. v. Vliet, "IS Architecture Complexity Dynamics in M&A: Does Consolidation Reduce Complexity?" *ECIS 2023 Research Papers*, May 2023. [Online]. Available: https://aisel.aisnet.org/ecis2023_rp/377

[19] E. Jochemsen, B. van den Hooff, M. Plomp, M. Rezazade Mehrizi, and K. Mol, "Shifting Complexity: The Impact of Modularization on IS Complexity: ECIS 2022," *ECIS 2022 RESEARCH PAPERS*, 2022.

[20] M. E. Iacob, J. Monteban, M. Van Sinderen, E. Hegeman, and K. Bitaraf, "Measuring Enterprise Architecture Complexity," in *2018 IEEE 22nd International Enterprise Distributed Object Computing Workshop (EDOCW)*. Stockholm: IEEE, Oct. 2018. doi: 10.1109/EDOCW.2018.00026. ISBN 9781538641415 pp. 115–124. [Online]. Available: https://ieeexplore.ieee.org/document/8536112/

[21] L. Manzur, J. M. Ulloa, M. Sánchez, and J. Villalobos, "xArchiMate: Enterprise Architecture simulation, experimentation and analysis," *SIMULATION*, vol. 91, no. 3, pp. 276–301, Mar. 2015. doi: 10.1177/0037549715575188. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0037549715575188

[22] ISO, "ISO/IEC/IEEE 42010:2022," Nov. 2022. [Online]. Available: https://www.iso.org/standard/74393.html

[23] C. Perks and T. Beveridge, Eds., *Guide to Enterprise IT Architecture*, ser. Springer Professional Computing. New York, NY: Springer New York, 2004. ISBN 9780387951324. [Online]. Available: http://link.springer.com/10.1007/b98880

[24] T. O. Group, "TOGAF® Standard." [Online]. Available: https://pubs.opengroup.org/togaf-standard/adm/chap01.html

[25] M. Rohloff, "Business Oriented Development of the IT Landscape: Architecture design on a Large Scale," in *Managing Worldwide Operations and Communications with Information Technology*, 2007.

[26] M. Iacob, H. Jonkers, D. Quartel, H. Franken, and H. van den Berg, *Delivering Enterprise Architecture with TOGAF and ArchiMate*. Enschede BiZZdesign 2012, 2012. ISBN 9789079240005 OCLC: 1073562791.

[27] S. Reiff-Marganiec and M. Tilly, Eds., *Handbook of Research on Service-Oriented Systems and Non-Functional Properties: Future Directions*. IGI Global, 2012. ISBN 9781613504321 9781613504338. [Online]. Available: http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-61350-432-1

[28] D. Hammer, "The many aspects of an IT-architecture," in *Proceedings International Conference and Workshop on Engineering of Computer-Based Systems*. Monterey, CA, USA: IEEE Computer. Soc. Press, 1997. doi: 10.1109/ECBS.1997.581891. ISBN 9780818678899 pp. 304–311. [Online]. Available: http://ieeexplore.ieee.org/document/581891/

[29] A. Tiwana and B. Konsynski, "Complementarities Between Organizational IT Architecture and Governance Structure," *Information Systems Research*, vol. 21, no. 2, pp. 288–304, Jun. 2010. doi: 10.1287/isre.1080.0206. [Online]. Available: https://pubsonline.informs.org/doi/10.1287/isre.1080.0206

[30] M. Brückmann, K.-M. Schöne, S. Junginger, and D. Boudinova, "Evaluating Enterprise Architecture Management Initiatives – How to Measure and Control the Degree of standardization of an IT Landscape," 2009.

[31] E. Stoica, "The Complexity of an IT Landscape," University of Twente, Enschede, Research Topics Component.

[32] M. Fowler and J. Lewis, "Microservices," Mar. 2014. [Online]. Available: https://martinfowler.com/articles/microservices.html

[33] R. Wrembel, "Data Integration Revitalized: From Data Warehouse Through Data Lake to Data Mesh," in *Database and Expert Systems Applications*, C. Strauss, T. Amagasa, G. Kotsis, A. M. Tjoa, and I. Khalil, Eds. Cham: Springer Nature Switzerland, 2023, vol. 14146, pp. 3–18. ISBN 9783031398469 9783031398476. [Online]. Available: https://link.springer.com/10.1007/978-3-031-39847-6_1

[34] Armbrust, Michael, A. Fox, Armando, Griffith, Rean, Joseph, D. Anthony, R. Katz, R. H, A. Konwinski, Andrew, G. Lee, Gunho, Patterson, D. A, Rabkin, Ariel, Stoica, and Matei, "Above the Clouds: A Berkeley View of Cloud Computing," Jan. 2009.

[35] M. Schumacher, Ed., *Security patterns: integrating security and systems engineering*, ser. Wiley series in software design patterns. Chichester, England ; Hoboken, NJ: John Wiley & Sons, 2006. ISBN 9780470858844 OCLC: ocm61651876.

[36] R. Abraham, J. Schneider, and J. Vom Brocke, "Data governance: A conceptual framework, structured review, and research agenda," *International Journal of Information Management*, vol. 49, pp. 424–438, Dec. 2019. doi: 10.1016/j.ijinfomgt.2019.07.008. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0268401219300787

[37] M. Fowler, *Patterns of enterprise application architecture*, ser. The Addison-Wesley signature series. Boston: Addison-Wesley, 2003. ISBN 9780321127426

[38] R. Sweeney, Ed., *Achieving Service-Oriented Architecture: Applying an Enterprise Architecture Approach*, 1st ed. Wiley, Jan. 2012. ISBN 9780470604519 9781119200178. [Online]. Available: https://onlinelibrary.wiley.com/doi/book/10.1002/9781119200178

[39] A. Berson, *Client/server architecture*, 2nd ed., ser. McGraw-Hill series on computer communications. New York: McGraw-Hill, 1996. ISBN 9780070056640

[40] B. Michelson, "Event-Driven Architecture Overview," Feb. 2006. [Online]. Available: http://www.customers.com/articles/event-driven-architecture-overview/

[41] S. R. Gardner, "Building the data warehouse," *Communications of the ACM*, vol. 41, no. 9, pp. 52–60, Sep. 1998. doi: 10.1145/285070.285080. [Online]. Available: https://dl.acm.org/doi/10.1145/285070.285080

[42] A. Nambiar and D. Mundra, "An Overview of Data Warehouse and Data Lake in Modern Enterprise Data Management," *Big Data and Cognitive Computing*, vol. 6, no. 4, p. 132, Nov. 2022. doi: 10.3390/bdcc6040132. [Online]. Available: https://www.mdpi.com/2504-2289/6/4/132

[43] C. Giebler, C. Gröger, E. Hoos, R. Eichler, H. Schwarz, and B. Mitschang, "The Data Lake Architecture Framework," 2021. doi: 10.18420/BTW2021-19. [Online]. Available: http://dl.gi.de/handle/20.500.12116/35802

[44] I. A. Machado, C. Costa, and M. Y. Santos, "Data Mesh: Concepts and Principles of a Paradigm Shift in Data Architectures," *Procedia Computer Science*, vol. 196, pp. 263–271, 2022. doi: 10.1016/j.procs.2021.12.013. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1877050921022365

[45] P. Erdi, *Complexity Explained*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. ISBN 9783540357773 9783540357780. [Online]. Available: http://link.springer.com/10.1007/978-3-540-35778-0

[46] S. Lloyd, "Measures of complexity: a nonexhaustive list," *IEEE Control Systems*, vol. 21, no. 4, pp. 7–8, Aug. 2001. doi: 10.1109/MCS.2001.939938. [Online]. Available: https://ieeexplore.ieee.org/document/939938/

[47] C. Schmidt, "Business Architecture Quantified: How to Measure Business Complexity," in *Business Architecture Management*, D. Simon and C. Schmidt, Eds. Cham: Springer International Publishing, 2015, pp. 243–268. ISBN 9783319145709 9783319145716. [Online]. Available: https://link.springer.com/10.1007/978-3-319-14571-6_13

[48] S. Morcov, L. Pintelon, and R. Kusters, "Definitions, characteristics and measures of IT project complexity - a systematic literature review," *International Journal of Information Systems and Project Management*, vol. 8, no. 2,

pp. 5–21, Oct. 2021. doi: 10.12821/ijispm080201. [Online]. Available: https://revistas.uminho.pt/index.php/ijispm/article/view/3581

[49] W. Weaver, "Science and Complexity," *American Scientist*, vol. 36, no. 4, pp. 536–544, 1948. [Online]. Available: https://www.jstor.org/stable/27826254

[50] P. D. Grünwald and P. M. Vitányi, "ALGORITHMIC INFORMATION THEORY," in *Philosophy of Information*. Elsevier, 2008, pp. 281–317. ISBN 9780444517265. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/B9780444517265500133

[51] G. Fioretti and B. Visser, "A Cognitive Approach to Organizational Complexity," *SSRN Electronic Journal*, 2004. doi: 10.2139/ssrn.524702. [Online]. Available: http://www.ssrn.com/abstract=524702

[52] M. Reeves, S. Levin, T. Fink, and A. Levina, "Taming Complexity," *Harvard Business Review*, Jan. 2020. [Online]. Available: https://hbr.org/2020/01/taming-complexity

[53] I. Gorzeń-Mitka and M. Okręglicka, "Managing Complexity: A Discussion of Current Strategies and Approaches," *Procedia Economics and Finance*, vol. 27, pp. 438–444, 2015. doi: 10.1016/S2212-5671(15)01018-7. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2212567115010187

[54] M. Paulk, B. Curtis, M. Chrissis, and C. Weber, "Capability maturity model, version 1.1," *IEEE Software*, vol. 10, no. 4, pp. 18–27, Jul. 1993. doi: 10.1109/52.219617. [Online]. Available: http://ieeexplore.ieee.org/document/219617/

[55] J. Poeppelbuss, B. Niehaves, A. Simons, and J. Becker, "Maturity Models in Information Systems Research: Literature Search and Analysis," *Communications of the Association for Information Systems*, vol. 29, 2011. doi: 10.17705/1CAIS.02927. [Online]. Available: https://aisel.aisnet.org/cais/vol29/iss1/27

[56] H. Jonkers and M.-E. Iacob, "Performance and Cost Analysis of Service-Oriented Enterprise Architectures:," in *Global Implications of Modern Enterprise Information Systems*, A. Gunasekaran, Ed. IGI Global, 2009, pp. 49–73. ISBN 9781605661469 9781605661476. [Online]. Available: http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-60566-146-9.ch004

[57] M.-E. Iacob and H. Jonkers, "Quantitative Analysis of Enterprise Architectures," in *Interoperability of Enterprise Software and Applications*, D. Konstantas, J.-P. Bourrières, M. Léonard, and N. Boudjlida, Eds. London: Springer-Verlag, 2006, pp. 239–252. ISBN 9781846281518. [Online]. Available: http://link.springer.com/10.1007/1-84628-152-0_22

[58] L. R. Fabrigar, D. T. Wegener, R. C. MacCallum, and E. J. Strahan, "Evaluating the use of exploratory factor analysis in psychological research." *Psychological Methods*, vol. 4, no. 3, pp. 272–299, Sep. 1999. doi: 10.1037/1082-989X.4.3.272. [Online]. Available: https://doi.apa.org/doi/10.1037/1082-989X.4.3.272

[59] M. Aigner, "Selecting the top three elements," *Discrete Applied Mathematics*, vol. 4, no. 4, pp. 247–267, Aug. 1982. doi: 10.1016/0166-218X(82)90048-8. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/0166218X82900488

[60] M. Krynicki and L. Szczerba, "On simplicity of formulas," *Studia Logica*, vol. 49, no. 3, pp. 401–419, Sep. 1990. doi: 10.1007/BF00370372. [Online]. Available: http://link.springer.com/10.1007/BF00370372

[61] U. Böckenholt and D. R. Lehmann, "On the limits of research rigidity: the number of items in a scale," *Marketing Letters*, vol. 26, no. 3, pp. 257–260, Sep. 2015. doi: 10.1007/s11002-015-9373-y. [Online]. Available: http://link.springer.com/10.1007/s11002-015-9373-y

[62] E. R. Poort and H. Van Vliet, "RCDA: Architecting as a risk- and cost management discipline," *Journal of Systems and Software*, vol. 85, no. 9, pp. 1995–2013, Sep. 2012. doi: 10.1016/j.jss.2012.03.071. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0164121212000994

[63] A. Abran, "The Design of Software Measurement Methods," in *Software Metrics and Software Metrology*. IEEE, 2010, pp. 99–128. ISBN 9780470606827. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6381957

[64] I. Holub, "Methodology for Measuring the Complexity of Enterprise Information Systems," *Journal of Systems Integration*, vol. 7, pp. 34–53, Jul. 2016. doi: 10.20470/jsi.v7i3.260

[65] V. C. Storey, M. Kaul, and C. Woo, "A Framework for Managing Complexity in Information Systems," *J. Database Manage.*, vol. 28, no. 1, pp. 31–42, Jan. 2017. doi: 10.4018/JDM.2017010103 Place: USA Publisher: IGI Global. [Online]. Available: https://doi.org/10.4018/JDM.2017010103

[66] W. Daoudi, K. Doumi, and L. Kjiri, "Adaptive Enterprise Architecture: Complexity Metrics in a Mixed Evaluation Method," in *Enterprise Information Systems*, J. Filipe, M. Śmia\lek, A. Brodsky, and S. Hammoudi, Eds. Cham: Springer International Publishing, 2022. ISBN 978-3-031-08965-7 pp. 505–523.

[67] R. J. Wieringa, *Design Science Methodology for Information Systems and Software Engineering*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014. ISBN 9783662438381 9783662438398. [Online]. Available: https://link.springer.com/10.1007/978-3-662-43839-8

[68] A. Priya, "Case Study Methodology of Qualitative Research: Key Attributes and Navigating the Conundrums in Its Application," *Sociological Bulletin*, vol. 70, no. 1, pp. 94–110, Jan. 2021. doi: 10.1177/0038022920970318. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0038022920970318

[69] H. Snyder, "Literature review as a research methodology: An overview and guidelines," *Journal of Business Research*, vol. 104, pp. 333–339, Nov. 2019. doi: 10.1016/j.jbusres.2019.07.039. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0148296319304564

[70] P. Dash, "Analysis of Literature Review in Cases of Exploratory Research," *SSRN Electronic Journal*, 2019. doi: 10.2139/ssrn.3555628. [Online]. Available: https://www.ssrn.com/abstract=3555628

[71] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Technical report, EBSE Technical Report EBSE-2007-01, Tech. Rep., 2007. [Online]. Avail-

able: https://www.cs.auckland.ac.nz/~norsaremah/2007%20Guidelines%20for%20performing%20SLR%20in%20SE%20v2.3.pdf

[72] J. Webster and R. T. Watson, "Analyzing the Past to Prepare for the Future: Writing a Literature Review," *MIS Quarterly*, vol. 26, no. 2, pp. xiii–xxiii, 2002. [Online]. Available: https://www.jstor.org/stable/4132319

[73] J. F. Wolfswinkel, E. Furtmueller, and C. P. M. Wilderom, "Using grounded theory as a method for rigorously reviewing literature," *European Journal of Information Systems*, vol. 22, no. 1, pp. 45–55, Jan. 2013. doi: 10.1057/ejis.2011.51. [Online]. Available: https://www.tandfonline.com/doi/full/10.1057/ejis.2011.51

[74] M. J. Page, J. E. McKenzie, P. M. Bossuyt, I. Boutron, T. C. Hoffmann, C. D. Mulrow, L. Shamseer, J. M. Tetzlaff, E. A. Akl, S. E. Brennan, R. Chou, J. Glanville, J. M. Grimshaw, A. Hróbjartsson, M. M. Lalu, T. Li, E. W. Loder, E. Mayo-Wilson, S. McDonald, L. A. McGuinness, L. A. Stewart, J. Thomas, A. C. Tricco, V. A. Welch, P. Whiting, and D. Moher, "The PRISMA 2020 statement: An updated guideline for reporting systematic reviews," *PLOS Medicine*, vol. 18, no. 3, p. e1003583, Mar. 2021. doi: 10.1371/journal.pmed.1003583. [Online]. Available: https://dx.plos.org/10.1371/journal.pmed.1003583

[75] D. R. Hofstadter, *Godel, Escher, Bach: an eternal golden braid*, 20th ed. New York: Basic Books, 1999. ISBN 9780394756820 9780465026562

[76] R. M. Gray, *Entropy and Information Theory*. Boston, MA: Springer US, 2011. ISBN 9781441979698 9781441979704. [Online]. Available: https://link.springer.com/10.1007/978-1-4419-7970-4

[77] C. Mesjasz, "Complexity of Social Systems," *Acta Physica Polonica A*, vol. 117, no. 4, pp. 706–715, Apr. 2010. doi: 10.12693/APhysPolA.117.706. [Online]. Available: http://przyrbwn.icm.edu.pl/APP/PDF/117/a117z468.pdf

[78] A. Santana, K. Fischbach, and H. Moura, "Enterprise Architecture Analysis and Network Thinking: A Literature Review," in *2016 49th Hawaii International Conference on System Sciences (HICSS)*. Koloa, HI, USA: IEEE, Jan. 2016. doi: 10.1109/HICSS.2016.567. ISBN 9780769556703 pp. 4566–4575. [Online]. Available: http://ieeexplore.ieee.org/document/7427753/

[79] J. Beese, S. Aier, K. Haki, and P. A. Khosroshahi, "DRIVERS AND EFFECTS OF INFORMATION SYSTEMS ARCHITECTURE COMPLEXITY: A MIXED-METHODS STUDY," *Research Papers*, Jun. 2016. [Online]. Available: https://aisel.aisnet.org/ecis2016_rp/74

[80] A. Schuetz, T. Widjaja, and R. Gregory, "Escape from Winchester Mansion – Toward a Set of Design Principles to Master Complexity in IT Architectures," in *International Conference on Information Systems (ICIS 2013): Reshaping Society Through Information Systems Design*, vol. 2, Dec. 2013.

[81] T. Widjaja, R. Gregory, and University of Virginia, USA, "Monitoring the Complexity of IT Architectures: Design Principles and an IT Artifact," *Journal of the Association for Information Systems*, vol. 21, no. 3, pp. 664–694, May 2020. doi: 10.17705/1jais.00616. [Online]. Available: https://aisel.aisnet.org/jais/vol21/iss3/4/

[82] A. W. Schneider, T. Reschenhofer, A. Schutz, and F. Matthes, "Empirical Results for Application Landscape Complexity," in *2015 48th Hawaii International Conference*

on System Sciences. HI, USA: IEEE, Jan. 2015. doi: 10.1109/HICSS.2015.490. ISBN 9781479973675 pp. 4079–4088. [Online]. Available: http://ieeexplore.ieee.org/document/7070309/

[83] R. Jain, A. Chandrasekaran, G. Elias, and R. Cloutier, "Exploring the Impact of Systems Architecture and Systems Requirements on Systems Integration Complexity," *IEEE Systems Journal*, vol. 2, no. 2, pp. 209–223, Jun. 2008. doi: 10.1109/JSYST.2008.924130. [Online]. Available: http://ieeexplore.ieee.org/document/4539770/

[84] K. Wehling and I. Schaefer, "Towards an Expert System for Identifying and Reducing Unnecessary Complexity of IT Architectures," 2017. [Online]. Available: https://dl.gi.de/items/44c546eb-bbfe-4b69-b269-d95913b91c0b

[85] I. Sommerville, D. Cliff, R. Calinescu, J. Keen, T. Kelly, M. Kwiatkowska, J. Mcdermid, and R. Paige, "Large-scale complex IT systems," *Communications of the ACM*, vol. 55, no. 7, pp. 71–77, Jul. 2012. doi: 10.1145/2209249.2209268. [Online]. Available: https://dl.acm.org/doi/10.1145/2209249.2209268

[86] M. Mocker, "What Is Complex About 273 Applications? Untangling Application Architecture Complexity in a Case of European Investment Banking," in *2009 42nd Hawaii International Conference on System Sciences*, Jan. 2009. doi: 10.1109/HICSS.2009.506 pp. 1–14, iSSN: 1530-1605. [Online]. Available: https://ieeexplore.ieee.org/document/4755701

[87] J. L. Lentz and T. M. Bleizeffer, "IT ecosystems: evolved complexity and unintelligent design," in *Proceedings of the 2007 symposium on Computer human interaction for the management of information technology*. Cambridge Massachusetts: ACM, Mar. 2007. doi: 10.1145/1234772.1234780. ISBN 9781595936356 p. 6. [Online]. Available: https://dl.acm.org/doi/10.1145/1234772.1234780

[88] S. Hacks, H. Hofert, J. Salentin, Y. C. Yeong, and H. Lichter, "Towards the Definition of Enterprise Architecture Debts," in *2019 IEEE 23rd International Enterprise Distributed Object Computing Workshop (EDOCW)*. Paris, France: IEEE, Oct. 2019. doi: 10.1109/EDOCW.2019.00016. ISBN 9781728145983 pp. 9–16. [Online]. Available: https://ieeexplore.ieee.org/document/8907262/

[89] N. Zazworka, C. Seaman, and F. Shull, "Prioritizing design debt investment opportunities," in *Proceedings of the 2nd Workshop on Managing Technical Debt*. Waikiki, Honolulu HI USA: ACM, May 2011. doi: 10.1145/1985362.1985372. ISBN 9781450305860 pp. 39–42. [Online]. Available: https://dl.acm.org/doi/10.1145/1985362.1985372

[90] L. Atchison, *Overcoming IT complexity: simplify operations, enable innovation, and cultivate successful cloud outcomes*. Cambridge: O'Reilly, 2023. ISBN 9781492098492 OCLC: on1351696495.

[91] A. Martini, J. Bosch, and M. Chaudron, "Architecture Technical Debt: Understanding Causes and a Qualitative Model," in *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications*. Verona, Italy: IEEE, Aug. 2014. doi: 10.1109/SEAA.2014.65. ISBN 9781479957958 pp. 85–92. [Online]. Available: http://ieeexplore.ieee.org/document/6928795/

[92] A. Martini and J. Bosch, "The Danger of Architectural Technical Debt: Contagious Debt and Vicious Circles," in *2015 12th Working IEEE/IFIP Conference on Software Architecture*. Montreal, QC, Canada: IEEE, May 2015. doi: 10.1109/WICSA.2015.31. ISBN 9781479919222 pp. 1–10. [Online]. Available: http://ieeexplore.ieee.org/document/7158498/

[93] B. Tieu and S. Hacks, "Determining Enterprise Architecture Smells from Software Architecture Smells," in *2021 IEEE 23rd Conference on Business Informatics (CBI)*. Bolzano, Italy: IEEE, Sep. 2021. doi: 10.1109/CBI52690.2021.10064. ISBN 9781665420693 pp. 134–142. [Online]. Available: https://ieeexplore.ieee.org/document/9610644/

[94] S. Daoudi, M. Larsson, KTH Royal Institute of Technology, Stockholm, Sweden, S. Hacks, Stockholm University, Stockholm, Sweden, J. Jung, and Frankfurt University of Applied Sciences, Frankfurt am Main, Germany, "Discovering and Assessing Enterprise Architecture Debts," *Complex Systems Informatics and Modeling Quarterly*, no. 35, pp. 1–29, Jul. 2023. doi: 10.7250/csimq.2023-35.01. [Online]. Available: https://csimq-journals.rtu.lv/article/view/csimq.2023-35.01

[95] J. Saat, S. Aier, and B. Gleichauf, "Assessing the Complexity of Dynamics in Enterprise Architecture Planning – Lessons from Chaos Theory," *AMCIS 2009 Proceedings*, Jan. 2009. [Online]. Available: https://aisel.aisnet.org/amcis2009/808

[96] G. Roedler, D. Rhodes, H. Schimmoller, and C. Jones, "Systems Engineering Leading Indicators Guide, Version 2.0," Jun. 2010. [Online]. Available: https://dspace.mit.edu/handle/1721.1/81893

[97] O. González-Rojas, A. López, and D. Correal, "Multilevel complexity measurement in enterprise architecture models," *International Journal of Computer Integrated Manufacturing*, vol. 30, no. 12, pp. 1280–1300, Dec. 2017. doi: 10.1080/0951192X.2017.1307453. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/0951192X.2017.1307453

[98] "EA Smells – Enterprise Architecture Debts," Aug. 2023. [Online]. Available: https://ea-debts.org/concept/ea-smells/

[99] J. Miguens, M. M. Da Silva, and S. Guerreiro, "A Viewpoint for Integrating Costs in Enterprise Architecture," in *On the Move to Meaningful Internet Systems. OTM 2018 Conferences*, H. Panetto, C. Debruyne, H. A. Proper, C. A. Ardagna, D. Roman, and R. Meersman, Eds. Cham: Springer International Publishing, 2018, vol. 11229, pp. 481–497. ISBN 9783030026097 9783030026103. [Online]. Available: https://link.springer.com/10.1007/978-3-030-02610-3_27

[100] R. Lagerström, P. Johnson, M. Ekstedt, U. Franke, and K. Shahzad, "Automated Probabilistic System Architecture Analysis in the Multi-Attribute Prediction Language (MAPL): Iteratively Developed using Multiple Case Studies," *Complex Systems Informatics and Modeling Quarterly*, no. 11, pp. 38–68, Jul. 2017. doi: 10.7250/csimq.2017-11.03. [Online]. Available: https://csimq-journals.rtu.lv/article/view/csimq.2017-11.03/975

[101] M.-E. Iacob, D. Quartel, and H. Jonkers, "Capturing Business Strategy and Value in Enterprise Architecture to Support Portfolio Valuation," in *2012 IEEE 16th International Enterprise Distributed Object Computing Conference*, 2012. doi: 10.1109/EDOC.2012.12 pp. 11–20.

[102] A. Aldea, M.-E. Iacob, M. Daneva, and L. H. Masyhur, "Multi-Criteria and Model-Based Analysis for Project Selection: An Integration of Capability-Based Planning, Project Portfolio Management and Enterprise Architecture," in *2019 IEEE 23rd International Enterprise Distributed Object Computing Workshop (EDOCW)*. Paris, France: IEEE, Oct. 2019. doi: 10.1109/EDOCW.2019.00032. ISBN 9781728145983 pp. 128–135. [Online]. Available: https://ieeexplore.ieee.org/document/8907326/

[103] M. Valja, N. Honeth, M. Buschle, R. Lagerstrom, K. K. Sasi, and S. Nithin, "An archimate based analysis of microgrid control system architectures," in *2014 International Conference on Embedded Systems (ICES)*. Coimbatore, India: IEEE, Jul. 2014. doi: 10.1109/EmbeddedSys.2014.6953179. ISBN 9781479950263 9781479950256 pp. 297–301. [Online]. Available: http://ieeexplore.ieee.org/document/6953179/

[104] P. Narman, T. Sommestad, S. Sandgren, and M. Ekstedt, "A framework for assessing the cost of IT investments," in *PICMET '09 - 2009 Portland International Conference on Management of Engineering & Technology*, 2009. doi: 10.1109/PICMET.2009.5262271 pp. 3154–3166.

[105] R. Lagerström, P. Johnson, and D. Höök, "Architecture analysis of enterprise systems modifiability – Models, analysis, and validation," *Journal of Systems and Software*, vol. 83, no. 8, pp. 1387–1403, Aug. 2010. doi: 10.1016/j.jss.2010.02.019. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0164121210000518

[106] R. Lagerström, P. Johnson, and M. Ekstedt, "Architecture analysis of enterprise systems modifiability: a metamodel for software change cost estimation," *Software Quality Journal*, vol. 18, no. 4, pp. 437–468, Dec. 2010. doi: 10.1007/s11219-010-9100-0. [Online]. Available: http://link.springer.com/10.1007/s11219-010-9100-0

[107] A. Abediniyan, M. Mohsenzadeh, and M. Abbasi Dezfouli, "A new Approach towards Cost and Benefit Enterprise Architecture Analysis," *International Journal of Computer Applications Technology and Research*, vol. 2, no. 2, pp. 160–165, Mar. 2013. doi: 10.7753/IJCATR0202.1015. [Online]. Available: http://www.ijcat.com/archives/volume2/issue2/ijcatr02021015.pdf

[108] L. Bai and M. Liu, "A semantics-supported XBRL model for enterprise total cost analysis system," in *2010 IEEE International Conference on Systems, Man and Cybernetics*, 2010. doi: 10.1109/ICSMC.2010.5642321 pp. 3294–3299.

[109] D. W. Swenson, "Managing costs through complexity reduction at Carrier Corporation," *Management Accounting (USA)*, vol. 79, no. 10, pp. 20+, Apr. 1998, section: 20. [Online]. Available: https://link.gale.com/apps/doc/A20878600/AONE?u=googlescholar&sid=bookmark-AONE&xid=6a576a30

[110] W. Perumal, "Complexity Costs." [Online]. Available: https://www.wilsonperumal.com/blog/blog/complexity-costs

[111] S. A. Wilson and A. Perumal, *Waging war on complexity costs: reshape your cost structure, free up cash flows, and boost productivity by attacking process, product and organizational complexity*. New York: McGraw-Hill, 2010. ISBN 9780071639132 OCLC: ocn472156182.

[112] L. da Fontoura Costa, "Quantifying Complexity (CDT-6)," May 2019. [Online]. Available: https://hal.science/hal-02126691

[113] D. J. Sturtevant, "System design and the cost of architectural complexity," Thesis, Massachusetts Institute of Technology, 2013. [Online]. Available: https://dspace.mit.edu/handle/1721.1/79551

[114] Y. Diao, A. Keller, S. S. Parekh, and V. Marinov, "Predicting Labor Cost through IT Management Complexity Metrics," *2007 10th IFIP/IEEE International Symposium on Integrated Network Management*, pp. 274–283, 2007. [Online]. Available: https://api.semanticscholar.org/CorpusID:14281091

[115] R. S. Dewi, A. P. Subriadi, and Sholiq, "A Modification Complexity Factor in Function Points Method for Software Cost Estimation Towards Public Service Application," *Procedia Computer Science*, vol. 124, pp. 415–422, 2017. doi: 10.1016/j.procs.2017.12.172. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S187705091732940X

[116] D. George, D. Gustafson, S. Henry, D. Hutchens, D. Kafura, and J. Sayler, "Predicting cost-of-change: from design structure metrics," *ACM SIGSOFT Software Engineering Notes*, vol. 7, no. 1, pp. 30–35, Jan. 1982. doi: 10.1145/1010809.1010814. [Online]. Available: https://dl.acm.org/doi/10.1145/1010809.1010814

[117] A. Georgiades, S. Sharma, T. Kipouros, and M. Savill, "Predicting and visualizing cost propagation due to engineering design changes," Aug. 2017.

[118] A. Banazadeh and M. H. Jafari, "A heuristic complexity-based method for cost estimation of aerospace systems," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 227, no. 11, pp. 1685–1700, Nov. 2013. doi: 10.1177/0954410012461987. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0954410012461987

[119] M. Murialdo and A. Cifuentes, "Quantifying Value with Effective Complexity," *Journal of Interdisciplinary Economics*, vol. 34, no. 1, pp. 69–85, Jan. 2022. doi: 10.1177/0260107920913663. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0260107920913663

[120] Z. Irani, A. Ghoneim, and P. E. Love, "Evaluating cost taxonomies for information systems management," *European Journal of Operational Research*, vol. 173, no. 3, pp. 1103–1122, Sep. 2006. doi: 10.1016/j.ejor.2005.07.007. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0377221705004893

[121] Y. Diao and K. Bhattacharya, "Estimating business value of IT services through process complexity analysis," in *NOMS 2008 - 2008 IEEE Network Operations and Management Symposium*, Apr. 2008. doi: 10.1109/NOMS.2008.4575136 pp. 208–215, iSSN: 2374-9709. [Online]. Available: https://ieeexplore.ieee.org/document/4575136/

[122] R. K. Yin, *Case study research and applications: design and methods*, sixth edition ed. Los Angeles: SAGE, 2018. ISBN 9781506336169

[123] W. Tellis, "Application of a Case Study Methodology," *The Qualitative Report*, vol. 3, no. 3, pp. 1–19, Sep. 1997. doi: 10.46743/2160-3715/1997.2015. [Online]. Available: https://nsuworks.nova.edu/tqr/vol3/iss3/1

[124] K. Devotta, J. Woodhall-Melnik, C. Pedersen, A. Wendaferew, T. P. Dowbor, S. J. Guilcher, S. Hamilton-Wright, P. Ferentzy, S. W. Hwang, and F. I. Matheson, "Enriching qualitative research by engaging peer interviewers: a case

study," *Qualitative Research*, vol. 16, no. 6, pp. 661–680, Dec. 2016. doi: 10.1177/1468794115626244. [Online]. Available: http://journals.sagepub.com/doi/10.1177/1468794115626244

[125] R. W. Oldendick, "Survey Research Ethics," in *Handbook of Survey Methodology for the Social Sciences*, L. Gideon, Ed. New York, NY: Springer New York, 2012, pp. 23–35. ISBN 9781461438755 9781461438762. [Online]. Available: https://link.springer.com/10.1007/978-1-4614-3876-2_3

[126] F. Scheuren, *What is a Survey?* American Statistical Association, 2004. [Online]. Available: https://books.google.nl/books?id=CPqsjgEACAAJ

[127] J. A. Krosnick, "Survey Research," *Annual Review of Psychology*, vol. 50, no. 1, pp. 537–567, Feb. 1999. doi: 10.1146/annurev.psych.50.1.537. [Online]. Available: https://www.annualreviews.org/doi/10.1146/annurev.psych.50.1.537

[128] R. C. Richey and J. D. Klein, *Design and Development Research: Methods, Strategies, and Issues*, 0th ed. Routledge, Jul. 2014. ISBN 9780203826034. [Online]. Available: https://www.taylorfrancis.com/books/9781136791291

[129] P. L. Flom, P. G. Supino, and N. P. Ross, "Constructing and Evaluating Self-Report Measures," in *Principles of Research Methodology*, P. G. Supino and J. S. Borer, Eds. New York, NY: Springer New York, 2012, pp. 147–175. ISBN 9781461433590 9781461433606. [Online]. Available: http://link.springer.com/10.1007/978-1-4614-3360-6_8

[130] D. A. Story and A. R. Tait, "Survey Research," *Anesthesiology*, vol. 130, no. 2, pp. 192–202, Feb. 2019. doi: 10.1097/ALN.0000000000002436. [Online]. Available: https://pubs.asahq.org/anesthesiology/article/130/2/192/20077/Survey-Research

[131] G. W. Ryan and H. R. Bernard, "Techniques to Identify Themes," *Field Methods*, vol. 15, no. 1, pp. 85–109, Feb. 2003. doi: 10.1177/1525822X02239569. [Online]. Available: http://journals.sagepub.com/doi/10.1177/1525822X02239569

[132] H. L. Ball, "Conducting Online Surveys," *Journal of Human Lactation*, vol. 35, no. 3, pp. 413–417, Aug. 2019. doi: 10.1177/0890334419848734. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0890334419848734

[133] J. Jamsen, K. Corley, J. Jamsen, and K. Corley, "E-Survey Methodology," Jan. 2001. [Online]. Available: https://www.igi-global.com/gateway/chapter/www.igi-global.com/gateway/chapter/20212

[134] L. Busetto, W. Wick, and C. Gumbinger, "How to use and assess qualitative research methods," *Neurological Research and Practice*, vol. 2, no. 1, p. 14, Dec. 2020. doi: 10.1186/s42466-020-00059-z. [Online]. Available: https://neurolrespract.biomedcentral.com/articles/10.1186/s42466-020-00059-z

[135] N. K. Denzin and Y. S. Lincoln, "Introduction: The Discipline and Practice of Qualitative Research." in *The Sage handbook of qualitative research, 3rd ed.* Thousand Oaks, CA: Sage Publications Ltd, 2005, pp. 1–32. ISBN 0-7619-2757-3 (Hardcover)

[136] D. A. Friedman, "How to Collect and Analyze Qualitative Data," in *Research Methods in Second Language Acquisition*, 1st ed., A. Mackey and S. M. Gass, Eds. Wiley, Nov. 2011, pp. 180–200. ISBN 9781444334272 9781444347340. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/9781444347340.ch10

[137] C. McMullin, "Transcription and Qualitative Methods: Implications for Third Sector Research," *VOLUNTAS: International Journal of Voluntary and Nonprofit Organizations*, vol. 34, no. 1, pp. 140–153, Feb. 2023. doi: 10.1007/s11266-021-00400-3. [Online]. Available: https://link.springer.com/10.1007/s11266-021-00400-3

[138] M. Naeem, W. Ozuem, K. Howell, and S. Ranfagni, "A Step-by-Step Process of Thematic Analysis to Develop a Conceptual Model in Qualitative Research," *International Journal of Qualitative Methods*, vol. 22, p. 16094069231205789, Jan. 2023. doi: 10.1177/16094069231205789. [Online]. Available: http://journals.sagepub.com/doi/10.1177/16094069231205789

[139] "What Is Spaghetti Architecture and How To Avoid It?" Feb. 2022. [Online]. Available: https://data-sleek.com/what-is-spaghetti-architecture-and-how-to-avoid-it/

[140] R. M. J.L, "Role of Middleware for Interoperability in Enterprise Architecture," University of Twente, May 2023.

[141] G. Hohpe and B. Woolf, *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. USA: Addison-Wesley Longman Publishing Co., Inc., Oct. 2003. ISBN 9780321200686

[142] M. E. Conway, "How do Committees Invent?" Datamation, Apr. 1968. [Online]. Available: https://www.melconway.com/Home/Committees_Paper.html

[143] J. Pollack, J. Helm, and D. Adler, "What is the Iron Triangle, and how has it changed?" *International Journal of Managing Projects in Business*, vol. 11, no. 2, pp. 527–547, May 2018. doi: 10.1108/IJMPB-09-2017-0107. [Online]. Available: https://www.emerald.com/insight/content/doi/10.1108/IJMPB-09-2017-0107/full/html

[144] B. Minto, "MECE: I invented it, so I get to say how to pronounce it." [Online]. Available: https://www.mckinsey.com/alumni/news-and-events/global-news/alumni-news/barbara-minto-mece-i-invented-it-so-i-get-to-say-how-to-pronounce-it

[145] Bizzdesign, "Metrics in Enterprise Studio." [Online]. Available: https://help.bizzdesign.com/articles/#!horizzon-help/metrics-in-enterprise-studio

[146] J. Baker, "The Technology–Organization–Environment Framework," in *Information Systems Theory*, Y. K. Dwivedi, M. R. Wade, and S. L. Schneberger, Eds. New York, NY: Springer New York, 2012, vol. 28, pp. 231–245. ISBN 9781441961075 9781441961082. [Online]. Available: https://link.springer.com/10.1007/978-1-4419-6108-2_12

[147] X. Wang, N. Guarino, G. Guizzardi, and J. Mylopoulos, "Towards an Ontology of Software: a Requirements Engineering Perspective," in *Frontiers in Artificial Intelligence and Applications*, vol. 267, Sep. 2014. doi: 10.3233/978-1-61499-438-1-317

[148] D. Petri, L. Mari, and P. Carbone, "A Structured Methodology for Measurement Development," *IEEE Transactions on Instrumentation and Measurement*, vol. 64, no. 9, pp. 2367–2379, Sep. 2015. doi: 10.1109/TIM.2015.2399023. [Online]. Available: http://ieeexplore.ieee.org/document/7050295/

[149] K. Gericke, C. Eckert, and M. Stacey, "Elements of a design method – a basis for describing and evaluating design methods," *Design Science*, vol. 8, p. e29, 2022. doi: 10.1017/dsj.2022.23. [Online]. Available: https://www.cambridge.org/core/product/identifier/S2053470122000233/type/journal_article

[150] L. Mari, M. Wilson, and A. Maul, "Fundamental Concepts in Measurement," in *Measurement Across the Sciences*. Cham: Springer International Publishing, 2023, pp. 19–48. ISBN 9783031224478 9783031224485. [Online]. Available: https://link.springer.com/10.1007/978-3-031-22448-5_2

[151] L. Richardson and M. Amundsen, *RESTful Web APIs*. Sebastopol, Calif.: O'Reilly, 2013. ISBN 9781449358068 OCLC: 863645887.

[152] R. Tietzmann, E. C. Pellanda, and A. F. Pase, "Application Programming Interface," in *The SAGE International Encyclopedia of Mass Media and Society*. Thousand Oaks,: SAGE Publications, Inc., Jan. 2020. [Online]. Available: https://sk.sagepub.com/reference/the-sage-encyclopedia-of-mass-media-and-society

[153] D. Loshin, *Master data management*. Amsterdam ; Boston: Elsevier/Morgan Kaufmann, 2009. ISBN 9780123742254

[154] A. Cleven and F. Wortmann, "Uncovering Four Strategies to Approach Master Data Management," in *Proceedings of the 2010 43rd Hawaii International Conference on System Sciences*, ser. HICSS '10. USA: IEEE Computer Society, Jan. 2010. doi: 10.1109/HICSS.2010.488. ISBN 9780769538693 pp. 1–10. [Online]. Available: https://doi.org/10.1109/HICSS.2010.488

[155] J. S. Addicks and P. Gringel, "Application Landscape Metrics: Overview, Classification, and Practical Usage." in *Enterprise Modelling and Information Systems Architectures, EMISA 2009*, Jan. 2009, pp. 55–68.

[156] C. Britton and P. Bye, *IT architectures and middleware: strategies for building large, integrated systems*, 2nd ed. Boston: Addison-Wesley, 2004. ISBN 9780321246943

[157] J. Lakhrouit and K. Baina, "Evaluating complexity of enterprise architecture components landscapes," in *2015 10th International Conference on Intelligent Systems: Theories and Applications (SITA)*. Rabat: IEEE, Oct. 2015. doi: 10.1109/SITA.2015.7358443. ISBN 9781509002207 pp. 1–5. [Online]. Available: http://ieeexplore.ieee.org/document/7358443/

[158] J. N. Landsberg, "Systems Architecting a Space Force Enterprise," Master's thesis, Massachusetts Institute of Technology, May 2022. [Online]. Available: https://dspace.mit.edu/bitstream/handle/1721.1/145164/landsberg-jnland-sm-sdm-2022-thesis.pdf?sequence=1

[159] D. Schattschneider, "Counting It Twice," *The College Mathematics Journal*, vol. 22, no. 3, pp. 203–211, May 1991. doi: 10.1080/07468342.1991.11973382. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/07468342.1991.11973382

[160] G. Blinowski, A. Ojdowska, and A. Przybylek, "Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation," *IEEE Access*, vol. 10, pp. 20 357–20 374, 2022. doi: 10.1109/ACCESS.2022.3152803. [Online]. Available: https://ieeexplore.ieee.org/document/9717259/

[161] I. Nadareishvili, R. Mitra, M. McLarty, and M. Amundsen, *Microservice Architecture: Aligning Principles, Practices, and Culture*, 1st ed. O'Reilly Media, Inc., Jul. 2016. ISBN 9781491956250

[162] C. Richardson, *Microservices patterns: with examples in Java*. Shelter Island, New York: Manning Publications, 2019. ISBN 9781617294549 OCLC: on1002834182.

[163] "*Pokémon Trading Card Game*," May 2024, page Version ID: 1225391666. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Pok%C3%A9mon_Trading_Card_Game&oldid=1225391666

[164] G. Guizzardi, "Ontology, Ontologies and the "I" of FAIR," *Data Intelligence*, vol. 2, no. 1-2, pp. 181–191. [Online]. Available: https://direct.mit.edu/dint/article/2/1-2/181/10008/Ontology-Ontologies-and-the-I-of-FAIR

[165] J. Moreira, "Semantic model-driven development for IoT interoperability of emergency services," PhD, University of Twente, Enschede, The Netherlands, Jul. 2019. [Online]. Available: http://purl.org/utwente/doi/10.3990/1.9789402815870

[166] C. L. Azevedo, M.-E. Iacob, J. P. A. Almeida, M. Van Sinderen, L. F. Pires, and G. Guizzardi, "Modeling resources and capabilities in enterprise architecture: A well-founded ontology-based proposal for ArchiMate," *Information Systems*, vol. 54, pp. 235–262, Dec. 2015. doi: 10.1016/j.is.2015.04.008. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0306437915000745

[167] N. Guarino, "Semantic matching: Formal ontological distinctions for information organization, extraction, and integration," in *Information Extraction A Multidisciplinary Approach to an Emerging Information Technology*, M. T. Pazienza, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, vol. 1299, pp. 139–170. ISBN 9783540634386 9783540695486. [Online]. Available: http://link.springer.com/10.1007/3-540-63438-X_8

[168] S. Matook and S. A. Brown, "Characteristics of IT artifacts: a systems thinking-based framework for delineating and theorizing IT artifacts," *Information Systems Journal*, vol. 27, no. 3, pp. 309–346, May 2017. doi: 10.1111/isj.12108. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1111/isj.12108

[169] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*. London England United Kingdom: ACM, May 2014. doi: 10.1145/2601248.2601268. ISBN 9781450324762 pp. 1–10. [Online]. Available: https://dl.acm.org/doi/10.1145/2601248.2601268

[170] A. Schütz, T. Widjaja, and J. Kaiser, *Complexity in Enterprise Architectures - Conceptualization and Introduction of a Measure from a System Theoretic Perspective*, Jan. 2013, journal Abbreviation: ECIS 2013 - Proceedings of the 21st European Conference on Information Systems Publication Title: ECIS 2013 - Proceedings of the 21st European Conference on Information Systems.

[171] J. Lakhrouit and K. Baina, "Evaluating enterprise architecture complexity using fuzzy AHP approach: Application to university information system," in *2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA)*. Marrakech, Morocco: IEEE, Nov. 2015. doi: 10.1109/AICCSA.2015.7507211. ISBN 9781509004782 pp. 1–7. [Online]. Available: http://ieeexplore.ieee.org/document/7507211/

[172] R. Lagerstrom, C. Baldwin, A. MacCormack, and S. Aier, "Visualizing and Measuring Enterprise Application Architecture: An Exploratory Telecom Case," in *2014 47th Hawaii International Conference on System Sciences*. Waikoloa,

HI: IEEE, Jan. 2014. doi: 10.1109/HICSS.2014.477. ISBN 9781479925049 pp. 3847–3856. [Online]. Available: http://ieeexplore.ieee.org/document/6759079/

[173] R. O. Stroud, A. Ertas, and S. Mengel, "Application of Cyclomatic Complexity in Enterprise Architecture Frameworks," *IEEE Systems Journal*, vol. 13, no. 3, pp. 2166–2176, Sep. 2019. doi: 10.1109/JSYST.2019.2897592. [Online]. Available: https://ieeexplore.ieee.org/document/8651315/

[174] B. Stroud and A. Ertas, "Enterprise cyclomatic complexity," in *2016 Annual IEEE Systems Conference (SysCon)*. Orlando, FL, USA: IEEE, Apr. 2016. doi: 10.1109/SYSCON.2016.7490520. ISBN 9781467395199 pp. 1–7. [Online]. Available: http://ieeexplore.ieee.org/document/7490520/

Appendices

Appendix A - IT Landscape Components in ArchiMate

In this appendix, the description of the IT landscape according to the prior work of the researcher is introduced. Figure A.1 is proposed by the researcher. According to the model, an IT landscape refers to the overarching structure, combination and configuration of applications (which can use several IT objects) and their relationships to support the business processes for delivering value to the customer.

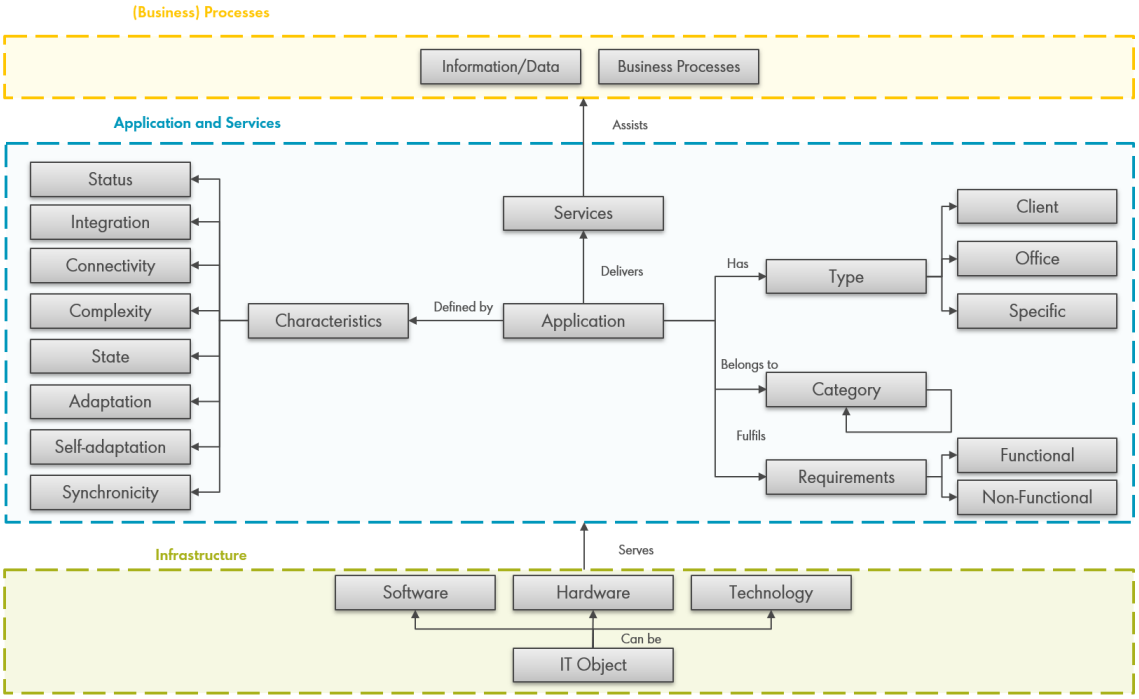


FIGURE A.1: IT Landscape Components

The model is comprised of three layers, similar to the core model of ArchiMate [1]. These are the business processes, applications and services, and the infrastructure layer.

The IT landscape is composed of a myriad of applications that support the business operations of an enterprise. The business has to complete certain processes, created based on information or data entities, that need to be supported by IT. Hence, from the business requirements, applications are selected and used by an organisation. These realise services that assist the business in delivering value to customers.

The applications can be combined and configured in various ways, and each of them has

certain characteristics. According to Matook [168], there are seven different characteristics of an application: integration, connectivity, complexity, state, adaptation, self-adaptation, and synchronicity. Furthermore, each application has a status or a lifecycle phase, as mentioned by Bruckmann et al. [30]. For example, an application can be in one of the following phases "proposed, test, productive, standard, and retired" [30].

Next, applications can have a type. For example, Laan [11] distinguished three types of applications: client (e.g., web browsers, word processors), office (e.g., email servers, collaboration tools), and business-specific (custom-built or highly customised applications, e.g., Enterprise Resource Planning). These applications can also belong to a category.

Moreover, the applications in an IT landscape fulfil certain functional or non-functional requirements [28]. The functional dimension is concerned with the high-level structure of architecture, thus, the applications, their specifications, interfaces or interaction sequences, protocols, data exchange formats and restrictions [28]. The non-functional dimension must investigate performance (e.g., response time, throughput), dependability (e.g., reliability, availability, security, and robustness) and general aspects (such as desirable future). This dimension is composed of requirements which were also perceived as important by Laan [11], as previously discussed.

The model illustrates that applications are using "IT objects" (as defined by Bruckmann), such as software (e.g., operating systems, databases, compilers, en-/decryption software), hardware (e.g., servers, workstations, storage devices) and technology. The last IT object, technology, is seen as "software architecture patterns" or "application protocols, digital preservation formats, programming languages" [30]. Matook [168] also supports the view that the "IT artifact" is based on technology, software, and hardware. These three elements are integrated, and they help with the execution of information-processing tasks with a specific usage purpose

Appendix B - Literature Review Protocol

In this appendix, information on the choices and actions undertaken during the literature reviews can be found. The details on why a certain type of literature review was performed to answer a particular question are introduced. Moreover, an in-depth delineation of the stages which guided the systematic literature review is discussed.

B.1 Knowledge Questions Break-down

Table B.1 illustrates how and why the sub-RQs were investigated in a particular manner. The sub-questions are the ones that created the holistic understanding on which the answer to the main question could be formed.

B.2 SLR Stages

To better explain the process behind the systematic literature review, this section entails the five SLR stages proposed by Webster and Watson [72].

Define

The first step of the literature review relates to clearly defining aspects such as the scope of the review or the data sources that will be used.

The scope of the review is obtaining answers to sub-RQs 1b and 2a. By having these well-defined research questions, through an SLR, they can progressively become answerable. The goal for sub-RQ 1b is to provide a background that can allow an appropriate positioning of future research activities, whereas sub-RQ 2a helps with identifying under-investigated areas of research.

For the outlet and the database selection, four academic repositories were used to perform the search. These are Scopus, IEEE Xplore, ISI Web of Science and ACM Digital Library.

Another important element in the "Define" phase is determining inclusion and exclusion criteria, as well as possible restrictions or limitations. Table B.2 depicts the choices for each of the two sub-RQs. These criteria aid with increasing the fit of the results and uncovering valuable insights.

Inc

For the inclusion criteria, the papers must be selected based on their relevance and the number of citations (peer-reviewed status). The language of the text must be English, as this is the primary language of the report. For both the research questions, the title, abstract and keywords of the paper must include various terms that are part of the text of the question. This is in place for a better alignment of the search result with the domain of interest. Moreover, as the document is concerned with the topic of IT complexity and EA is a means to manage it, the whole domain of Information Systems (IS), containing sub-domains as EA, must guide the inclusion of articles.

This has a direct impact on the exclusion criteria. The articles which are part of domains outside IS will be disregarded. For example, the domain of business management does not address IT architectural complexity but instead discusses business intricacies. On top of that, specifically for sub-RQ 1b, the origins of complexity should apply to the IT landscape (considering the connection between business – application – technology landscapes), hence, investigating government techniques or management practices, finding

complexity causes in supply chains or complexity emerging during software design will be excluded. These types of exclusions are done as the possible results narrow the perspective of the research and limit the findings to singular events, instead of observing the holistic view and intricacies of an enterprise.

Lastly, keywords or search terms were defined. The terms helped guide the search and set the basis for possible search queries. They are the ones encompassed in the formulation of the RQ or their synonyms and variations. For RQ 1b, Table B.3 illustrates the main keywords which form the core of RQ 1b. Table B.4 showcases the main keywords used for RQ 2a.

Search

The search phase is concerned with performing the search on the above-mentioned repositories (Scopus, IEEE Xplore, ISI Web of Science and ACM Digital Library). Based on the keywords, different queries were initially created and through episodic searches, they were altered such that sufficient and relevant outcomes could emerge. Wolfswinkel et al. mention that for the sake of transparency, it is important that the search terms, queries, and sources are documented [73]. Thus, the search queries applied for each of the scientific repositories, for sub-RQ 1b are visible in Table B.5. Table B.6 offers an overview of the queries for sub-RQ 2a.

Table B.1: Breakdown Research Question

Knowledge Question	Sub-Questions	Type of Literature Review	Motivation
Which complexity metrics can be used to measure the change in an IT landscape?	How can complexity be classified?	Exploratory	As complexity is a recurring topic in numerous research areas, discovering a common classification structure is essential. The body of knowledge on complexity is extensive, thus an exploratory literature review is more appropriate for creating a holistic understanding of the topic. Furthermore, with the help of an exploratory literature review, it can be uncovered which classification components are not thoroughly investigated.
	What factors drive/influence the complexity of an IT landscape?	Systematic	To identify which factors make an IT landscape complex, a systematic literature review was performed. By having such a structured approach, the most important causes/origins of complexity can be discovered.
	What are existing metrics operationalisations for measuring architectural complexity?	Exploratory	To understand what formulas can operationalise the origins of complexity, an exploratory literature review was performed to check existing complexity metrics and their measurements.
What types of costs are associated with the IT landscape?	What Enterprise Architecture model-based cost analysis techniques can be applied to IT landscapes?	Systematic	Various model-based cost analysis techniques exist. However, this question aims to search for those that can be applied to Enterprise Architecture models (e.g., such as ArchiMate representations), with the help of a systematic literature review. Different types of costs are expected to be illustrated in such models.
	What are the prevalent usages of "cost of complexity"?	Exploratory	The synthesis of the terms “cost” and “complexity” is desired to be investigated with the help of this question. By performing an exploratory literature review, the areas in which “the cost of complexity” is used can gain increased clarity, as well as discover what are common practices for calculating such costs.

Table B.2: Inclusion and Exclusion Criteria

Research Question	RQ 1b	Inclusion	Based on relevance
			Language - English
			Title, abstract and keywords – Include the term complexity or its derivates
			Thematic relevance – Fall under research areas concerned with information systems, enterprise architecture or computer science
		Exclusion	Articles from medicine, social sciences, business management, mathematics natural sciences, education
			Articles discussing IT projects, government, management, supply chain, industry-specific concerns or software design
	RQ 2a	Inclusion	Based on relevance
			Language – English
			Title, abstract and keywords – Include the terms related to models and costs applications
		Exclusion	Articles falling outside of the information system domain

Table B.3: RQ 1b Keywords

Complexity	Drivers	Domain of Interest
Complex*	Driver*	IT
	Factor*	IT architecture
	Cause*	IT landscape
	Root Cause*	Application* architecture
	Origin*	Application* landscape
	Source*	Enterprise architecture
	Influence*	Information system* architecture

Table B.4: RQ 2a Keywords

Model-based	Cost	Domain of Interest
Model-based	Cost*	Enterprise Architecture
Model	Cost-analys*	ArchiMate
Model-supported	Cost-estimation	Application* architecture
	Econom*	

Table B.5: RQ 1b Search Query

Main Search Queries	Scopus	General	(TITLE-ABS-KEY (("IT" OR "IT architecture" OR "IT landscape" OR "Application architecture" OR "Application landscape" OR "Enterprise architecture" OR "Enterprise landscape" OR "Information systems architecture") AND ("complexity" OR "Complexity driver" OR "driver* of complexity" OR "Complexity origin" OR "origin* of complexity" OR "Complexity factor" OR "factor* of complexity" OR "Complexity source" OR "source* of complexity" OR "Complexity cause" OR "cause* of complexity" OR "Complexity root cause" OR "Complexity influence" OR "influence* of complexity")) AND TITLE (complex*) AND KEY (it OR "information system architecture"))
	IEEE Xplore	Title	Complexit*
		Abstract	IT OR IT architecture OR IT landscape OR Application architecture OR Application landscape OR Enterprise architecture OR Enterprise landscape OR Information systems architecture
			Complexity driver OR driver* of complexity OR Complexity origin OR origin* of complexity OR Complexity factor OR factor* of complexity OR Complexity source OR source* of complexity OR Complexity cause OR cause* of complexity OR Complexity root cause OR Complexity influence OR influence* of complexity
	ACM Digital	General	("IT" OR "IT architecture" OR "IT landscape" OR "Application architecture" OR "Application landscape" OR "Enterprise architecture" OR "Enterprise landscape" OR "Information systems architecture") AND ("complexity" OR "Complexity driver" OR "driver* of complexity" OR "Complexity origin" OR "origin* of complexity" OR "Complexity factor" OR "factor* of complexity" OR "Complexity source" OR "source* of complexity" OR "Complexity cause" OR "cause* of complexity" OR "Complexity root cause" OR "Complexity influence" OR "influence* of complexity") AND TITLE (complex*)
	ISI Web of Science	Title	IT OR IT architecture OR IT landscape OR Application architecture OR Application landscape OR Enterprise architecture OR Enterprise landscape OR Information systems architecture
			Complexity driver OR driver* of complexity OR Complexity origin OR origin* of complexity OR Complexity factor OR factor* of complexity OR Complexity source OR source* of complexity OR Complexity cause OR cause* of complexity OR Complexity root cause OR Complexity influence OR influence* of complexity

Table B.6: RQ 2a Search Query

Main Search Queries	Scopus	General	(TITLE-ABS-KEY ("Application architecture" OR "Enterprise architecture" OR "ArchiMate") AND ("Model-based" OR "Model based" OR "Model supported") AND ("cost*" OR "econom*" OR "cost-analysis" OR "cost estimation"))
	IEEE Xplore	Title	Model* OR Cost*
		Abstract	Model-based OR Model based OR Model supported
			Cost* OR Cost-analysis OR Cost-estimation
			Enterprise Architecture OR ArchiMate OR Application Architecture
	ACM Digital	General	(""Enterprise architecture" OR ”Application Architecture" OR ”ArchiMate") AND ("cost*" OR "cost-analysis" OR "cost estimation”) AND (“Model-based” OR “Model Supported”)
	ISI Web of Science	Title	Model* OR Cost*
		Topic	Enterprise Architecture OR ArchiMate OR Application Architecture
			Model-based OR Model based OR Model supported
			Cost* OR Cost-analysis OR Cost-estimation

Select

In the "Select" phase, from the number of articles discovered, Wolfswinkel et al. mention that the reviewer must filter out the doubles, refine samples based on the abstract and title, then consider the full text, and lastly perform forward and backward citation searches [73]. For RQ 1b, 10 papers were reviewed, given that they fell under the scope of the research. Initially, the queries returned 173, 64, 89 and 91 results for Scopus, IEEE Xplore, ACM Digital Library, and ISI Web of Science. Moving to RQ 2a, 7 papers were selected from 37, 18, 82 and 167 results (Scopus, IEEE Xplore, ACM Digital Library, and ISI Web of Science). By filtering out the doubles, analysing the titles and abstracts of the results, as well as consulting the full text, the final selection of academic work has been made.

To ensure that the search results are as comprehensive as possible, a snowballing technique has been used [169], which is in line with performing a forward and backward citation, suggested by Wolfswinkle et al. [73]. By looking at the references used by the authors, as well as where the paper is being cited, additional papers with new points of view can be analysed. For question 1b, only one new insight could be gathered in terms of complexity drivers as many of the papers were referencing each other. However, a lot of the authors were touching upon factors which could be classified as architectural debt, thus 6 additional papers on this topic were investigated as possible drivers for complexity.

The results and the citations of the 17 papers (complexity origins:11 and architectural debt:6) pointed to insights for sub-RQ 1b, discussing metrics which could be useful in the calculation of complexity. For RQ 2a, by applying the snowball technique, 4 additional papers were inspected as they were closely linked with the question. An overview of the numbers described above is visible in Figure B.1 and B.2 below, in line with the PRISMA methodology [74].

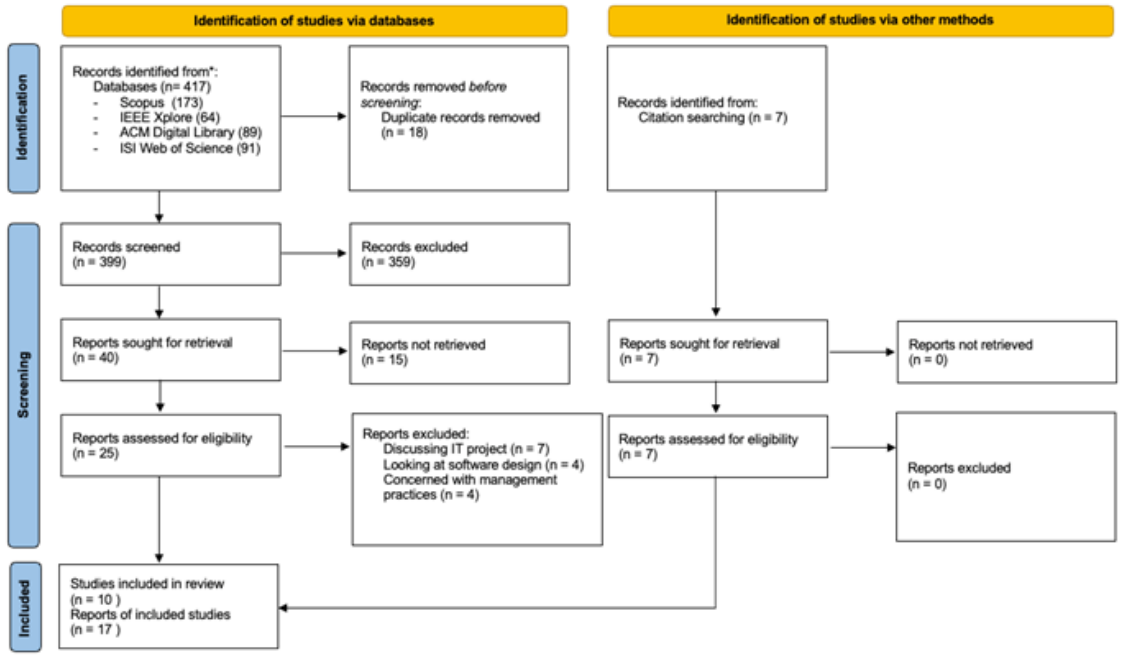


FIGURE B.1: PRISMA - Complexity Metrics (RQ 1b)

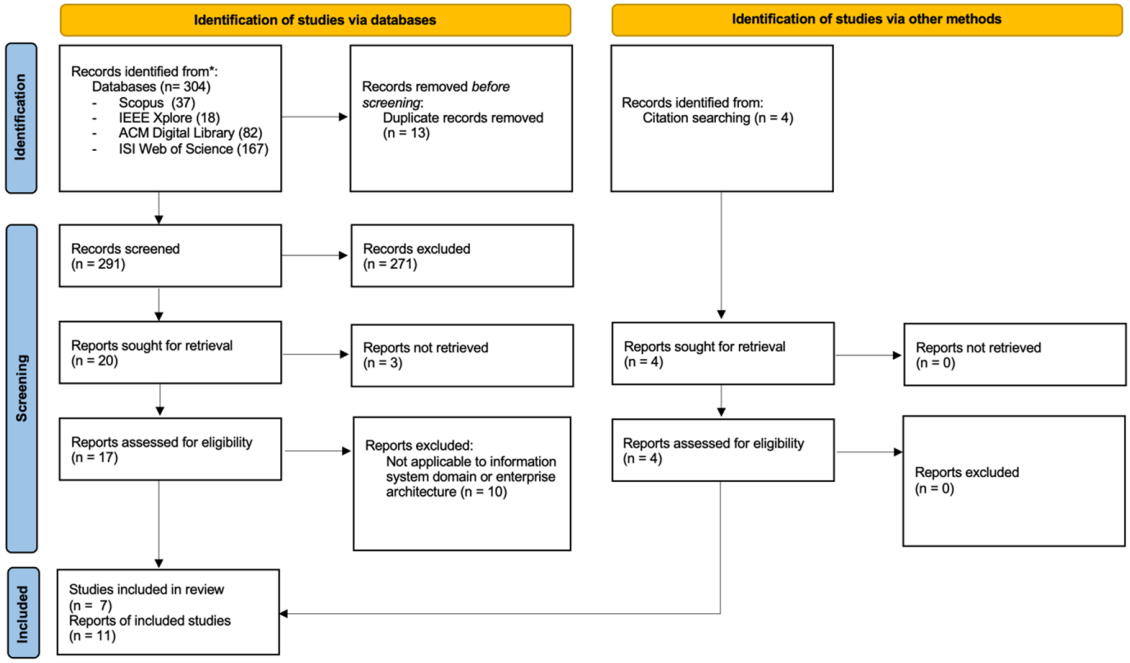


FIGURE B.2: PRISMA - Costs (RQ 2a)

Analyse

Moving to the analysis phase of the five-stage method [73], the authors explain that from the pool of selected papers, a random one will be chosen as a starting point and concepts that seem relevant must be highlighted throughout the paper. All the academic articles up for review will undergo this process once. The reviewer will thus perform open-coding (to discover hidden aspects from a bird's eye perspective of the study), axial coding (to find relationships between different categories discovered) and finally selective coding (to integrate and refine different categories).

Furthermore, the authors describe how important is to create a concept matrix for reviewing literature. Thus, instead of categorising the article itself, find and categorise key concepts. Table B.7 and B.8 showcase the basic concept matrix for the concepts uncovered for sub-RQ 1b, whereas Table B.9 illustrates the topics discovered for question 2a.

Table B.7: Concept Matrix Complexity Drivers

Article	Author	Concepts				
		Complexity Metrics	IT / Application architecture	Business Support	Drivers of Complexity	Unintelligent design
Decision support for reducing unnecessary IT complexity of application architectures	K. Wehling, D. Wille, C. Seidl, and I. Schaefer	X	X	X		
Empirical results for application landscape complexity	A. W. Schneider, T. Reschenhofer, A. Schutz, and F. Matthes	X	X		X	
Drivers and Effects of Information Systems Architecture Complexity: A Mixed-Methods Study	R. Beese, M. Kazem, and A. Khosroshahi	X	X		X	
Escape from Winchester Mansion-toward a Set of Design Principles to Master Complexity in IT Architectures	A. Schütz, T. Widjaja, and R. W. Gregory		X		X	
Monitoring the complexity of IT architectures: Design principles and an IT artifact	T. Widjaja and R. W. Gregory		X		X	
Towards an expert system for identifying and reducing unnecessary complexity of IT architectures	K. Wehling and I. Schaefer	X	X			
Large-scale Complex IT Systems	I. Sommerville				X	
What Is Complex Abput 273 Applications? Untangling Application Architecture Complexity in a Case of European Investment Banking	M. Mocker	X	X	X		
Exploring the Impact of Systems Architecture and Systems Requirements on Systems Integration Complexity	R. Jain, A. Chandrasekaran, G. Elias, and R. Cloutier				X	
IT Ecosystems: Evolved Complexity and Unintelligent Design	J. L. Lentz and T. M. Bleizeffer				X	X

Table B.8: Concept Matrix Architectural Debt

Article	Author	Concepts					
		Technical Debt	Architectur. Technical Debt	Causes of Architectural Debt	Emerging Issues Due to Debt	Enterprise Architecture Debt	EA Smells
Towards the definition of enterprise architecture debts	S. Hacks, H. Hofert, J. Salentin, Y. C. Yeong, and H. Lichter		X		X	X	
Prioritizing design debt investment opportunities	N. Zazworka, C. Seaman, and F. Shull	X			X		
Architecture technical debt: Understanding causes and a qualitative model	A. Martini, J. Bosch, and M. Chaudron		X	X			
The Danger of Architectural Technical Debt: Contagious Debt and Vicious Circles	A. Martini and J. Bosch		X	X	X		
Determining Enterprise Architecture Smells from Software Architecture Smells	B. Tieu and S. Hacks			X		X	X
Discovering and Assessing Enterprise Architecture Debts	S. Daoudi, M. Larsson, S. Hacks, and J. Jung			X		X	

Table B.9: Concept Matrix Model-Based Cost Analysis

Article	Author	Concepts						
		ArchiMate	(Time-based) Activity Costing	Top and Bottom-up approaches	Cost, risk and benefits EA model-based	Project oriented	Modifiability	Final Value
A Viewpoint for Integrating Costs in Enterprise Architecture	J. Miguens, M. M. da Silva, and S. Guerreiro	X	X	X	X			
Performance and Cost Analysis of Service-Oriented Enterprise Architectures	H. Jonkers and M.-E. Iacob	X		X	X			
Quantitative Analysis of Enterprise Architectures	M.-E. Iacob and H. Jonkers	X		X				X
Multi-Criteria and Model-Based Analysis for Project Selection: An Integration of Capability-Based Planning, Project Portfolio Management and Enterprise Architecture	A. Aldea, M.-E. Iacob, M. Daneva, and L. H. Masyhur					X	X	
From enterprise architecture to business models and back	M. E. Iacob, L. O. Meertens, H. Jonkers, D. A. C. Quartel, L. J. M. Nieuwenhuis, and M. J. van Sinderen	X			X			
An archimate based analysis of microgrid control system architectures	M. Valja, N. Honeth, M. Buschle, R. Lagerstrom, K. K. Sasi, and S. Nithin	X			X		X	
A framework for assessing the cost of IT investments	P. Narman, T. Sommestad, S. Sandgren, and M. Ekstedt	X	X		X	X	X	X
Architecture analysis of enterprise systems modifiability: a metamodel for software change cost estimation	R. Lagerström, P. Johnson, and M. Ekstedt				X	X	X	X
Architecture analysis of enterprise systems modifiability – Models, analysis, and validation	R. Lagerström, P. Johnson, and D. Höök		X		X	X	X	
A new Approach towards Cost and Benefit Enterprise Architecture Analysis	A. Abediniyan, M. Mohsenzadeh, and M. Abbasi Dezfouli					X		
A semantics-supported XBRL model for enterprise total cost analysis system	L. Bai and M. Liu					X		

Present

The last step of the literature review is concerned with presenting not only the findings of the corpus of papers but also the relationships between concepts and insights. Section 2.3 encompasses the answers for sub-RQ 1 and 2. Moreover, at the end of each section analysing the literature, a conclusion synthesising the findings is included. In this synthesis, graphical representations/illustrations/tables are provided such that the reader can be made aware of discovered connections. Having this step is also considered important by Wolfswinkel et al., as they state that "visualizations can help reach a broader audience"[73].

Appendix C - Complexity Metrics Overview

C.1 Documented Complexity Metrics

Figure C.1 illustrates the findings of Iacob et al. [20]. The 42 metrics for complexity and their classification according to the framework of Schneider is presented [2].

C.2 Extension for Documented Metrics

To extend the work of Iacob et al. [20] with recent literature developments, Table C.1 has been created.

Table C.1: Proposed Metrics for Complexity

Metric	Explaination	Operationalisation	Ref.
Entropy	For measuring the heterogeneity of components and relationships. pi is the relative frequency of a certain flavour i; e.g., the number of instances for operating system i. Use entropy on certain parts of EA; EA is a graph with nodes and the relationships between components are edges.	EM=-$\sum_{i=1}^{n} p_i \ln(p_i)$	[47], [170]
Complexity (e.g., Coupled Domain Complexity)	Complexity C is rooted in the number N and the heterogeneity H of components T and relations R. Calculate the number of interfaces of applications assigned to a functional domain (D) and the heterogeneity of the functional domains. Domains are coupled based on the information flows of applications which are part of a domain.	$C_x = (N_x,\ H_x),\ x \in T, R$	[82]

Continued on next page

Table C.1: Proposed Metrics for Complexity (Continued)

Metric	Explaination	Operationalisation	Ref.
Fuzzy AHP	Multiple criteria decision-making tool for to-be scenarios. Check density of components from each layer – business, application, technology. Give weights (weight vector calculated based on expert's ratings) to both the layers and the individual components. Choose the minimum density architecture.	V=[B$_1$, B_2, A_1, A_2, T_1, T_2] $L = [L_1,\ L_2,\ L_3]$ $W =$ $[W_1,\ W_2,\ W_3,\ W_4,\ W_5,\ W_6\ ]$ $B = B_1 * W_{11} + B_2 * W_{12}$ $A = A_1 * W_{21} + A_2 * W_{22}$ $T = T_1 * W_{31} + T_2 * W_{32}$ $C = L_1 * B + L_2 * A + L_3 * T$	[157], [171]
Coupling Metric	The manner and degree of interdependence between software modules. i is the level of highest coupling type and n is the number of dependencies between two components.	C= i+n/n+1	[172]
Design Structure Matrix – Visibility and Propagation	A method for uncovering new facts about applications and architectures based on relationships. Based on a design structure matrix calculate how many applications depend directly or indirectly on one application (A) – Visibility Fan In, and how many A depends on – Visibility Fan Out.	Propagation Cost=$\sum_{i=1}^{N} VFI_i\ /N^2 = \frac{\sum_{i=1}^{N} VFO_i}{N^2}$	[172]
(Enterprise) Cyclomatic Complexity	Verticles, edges, and connected components.	V= e-n+2p	[172], [173], [174]
Structural Complexity Metric	As explained in 2.3.1	S C M=F$^{3.11}$ + $D^{3.11}$	[97]

Continued on next page

Table C.1: Proposed Metrics for Complexity (Continued)

Metric	Explaination	Operationalisation	Ref.
EA Degree of Dynamic Complexity	As explained in 2.3.1	DDC i, i+1(t)=$\sum_{j=1}^{n} \frac{fj(t)+fj}{n}$, where $n \in N$	[66]

Metric	Objective/ subjective	Structural/ dynamic	Quantitative /qualitative	Ordered/ disordered
# relations	Objective	Structural	Quantitative	Ordered
# elements	Objective	Structural	Quantitative	Ordered
# cardinal elements	Objective	Structural	Quantitative	Ordered
# cardinal relations	Objective	Structural	Quantitative	Ordered
Cyclomatic complexity	Objective	Structural	Quantitative	Ordered
Element entropy	Objective	Structural	Quantitative	Disordered
Relation entropy	Objective	Structural	Quantitative	Disordered
Conformity	Objective	Structural	Quantitative, qualitative	Disordered
Interface Complexity Multiplier	Objective	Structural, dynamic	Quantitative, qualitative	Ordered, disordered
Redundancy	Objective	Structural	Quantitative, qualitative	Disordered
# OS & middleware	Objective	Structural	Quantitative	Ordered
Functions/system	Objective	Structural	Quantitative	Ordered
# patterns	Objective	Structural	Quantitative	Ordered
Application age	Objective	Structural	Quantitative	Ordered
# hardware platforms	Objective	Structural	Quantitative	Ordered
Betweenness centrality	Objective	Structural	Quantitative	Disordered
Quantified expert opinion	Subjective	Structural	Quantitative	Ordered
Pattern coverage	Objective	Structural	Quantitative	Ordered
Elements/type	Objective	Structural	Quantitative	Ordered
Relations/element	Objective	Structural	Quantitative	Ordered
Processes/element	Objective	Structural	Quantitative	Ordered
Elements/process	Objective	Structural	Quantitative	Ordered
Service-time Actual	Objective	Structural	Quantitative	Ordered
Domains/application	Objective	Structural	Quantitative	Ordered
Software categories/app	Objective	Structural	Quantitative	Ordered
SLOC	Objective	Structural	Quantitative	Ordered
Halstead difficulty	Objective	Structural	Quantitative	Ordered
# functions	Objective	Structural	Quantitative	Ordered
Apps/user	Objective	Structural	Quantitative	Ordered
Customization	Objective	Structural	Quantitative	Ordered
# instances	Objective	Structural	Quantitative	Ordered
# software platforms	Objective	Structural	Quantitative	Ordered
Application type	Objective	Structural	Qualitative	Ordered
# software frameworks	Objective	Structural	Quantitative	Ordered
# new applications	Objective	Structural	Quantitative	Ordered
# retired applications	Objective	Structural	Quantitative	Ordered
# physical servers	Objective	Structural	Quantitative	Ordered
# virtual servers	Objective	Structural	Quantitative	Ordered
Visibility Fan-In	Objective	Structural	Quantitative	Ordered
Visibility Fan-out	Objective	Structural	Quantitative	Ordered
Requirements/app	Objective	Structural	Quantitative	Ordered
Propagation cost	Objective	Structural	Quantitative	Disordered

FIGURE C.1: Complexity Metrics List

Appendix D - Survey and Interview Questions

D.1 Survey

Title Understanding the IT Application Landscape Complexity

Context For my master's thesis, I aim to create a method and measurement for quantifying Application Landscape complexity in large enterprises. Moreover, it is examined how complexity affects change processes. The main purpose of this survey is to get your input on the drivers and significance of complexity.

Section 1 – Demographic Information

All the collected information will be treated as confidential and handled with the utmost care. The name will only be used to discuss the answers further in the case of an interview.

1. What is your name? – This response will only be used for the identification of the answer

2. What is your position in the company?

    o IT manager
    o Lead Architect
    o Solution Architect
    o Data Architect
    o Other ...

Section 2 – Introduction to Complexity

3. Do you consider the IT Application Landscape of your company as being complex?

    o Yes
    o No

4. (If Yes) What are the elements in the Application Landscape that drive complexity?

5. (If No) What are the elements that make the Application Landscape simple?

6. What challenges arise due to the complexity of an Application Landscape? – You can select multiple options and you are also encouraged to name a few more via the "Other" prompt.

    o Costly to manage and operate the IT landscape
    o Less flexibility to future changes
    o Difficulties when including innovations
    o Other ...

7. Can complexity of an Application Landscape bring any benefits? If so, which are those?

Section 3 – Maturity of IT Landscape Complexity Management

8. Do you think that Application Landscape complexity is under control in your company?

    o Yes
    o No

9. Do you know initiatives or projects that are trying to simplify the Application Landscape in your company? - You can select multiple options and you are also encouraged to name a few more via the "Other" prompt.

- o Initiative 1
- o Initiative 2
- o Initiative 3
- o Initiative 4
- o Other ...

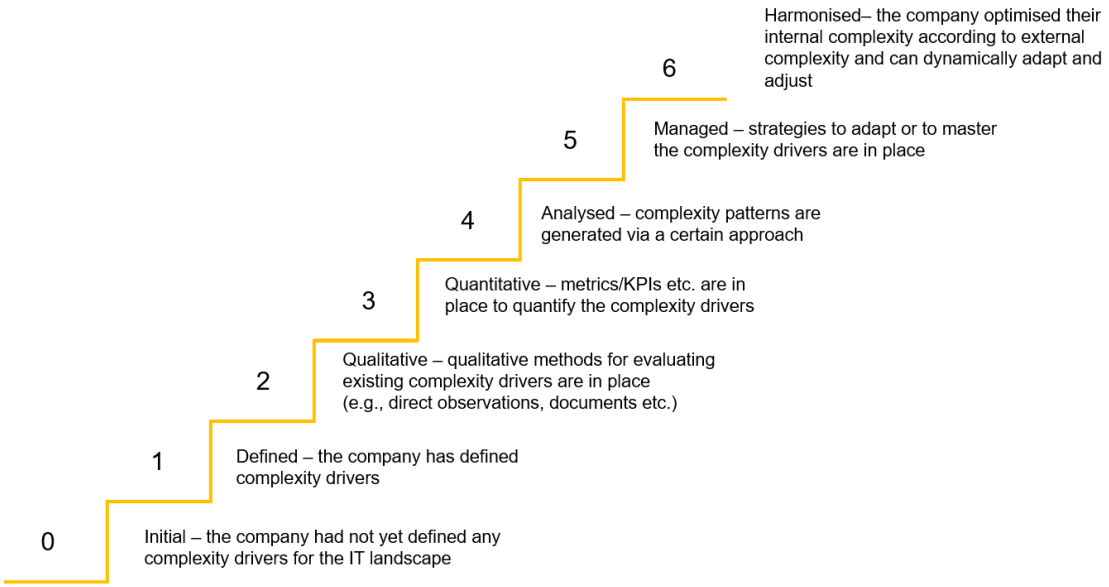
10. Is there a clear, standard definition used in the company for complexity factors which influence the Application Landscape?

- o Yes
- o No

11. Do you think a clear, standard definition is needed? Why?

12. On which level would you position your company when it comes to capabilities for IT landscape complexity management? - The figure below depicts a complexity capability maturity model. Level 0 is linked with a low level of maturity, and level 6 means the highest possibility for complexity management according to the literature. Please read it carefully and position your company on the stage that you consider appropriate.

- o Level 0
- o Level 1
- o Level 2
- o Level 3
- o Level 4
- o Level 5
- o Level 6



13. What existing information sources in your company can be used to gather quantitative details about complexity drivers? – Please name systems, document types or people/roles that can deliver this information

14. Are there any tools or methods that you know of which can be used to analyse and quantify the complexity of an Application Landscape?

15. (If yes) Please name the tool(s) or method(s).

Section 4 – Complexity Factors and Costs

16. What makes an IT landscape complex at an application/data level? - Please rank the items below how you see fit, by either drag and dropping or by using the arrows on the right hand side of the item.

o Number of components
o Integrations/Interfaces/Information flows between applications
o Customised applications
o Integration styles (e.g., bespoke, point-to-point solutions)
o Variety of constructing element of component (e.g., programming languages or different technologies)
o Overlap between data of applications
o Overlap between application functionality

o Lack of documentation

17. Are there other factors (besides the ones described above) that you consider essential, which can influence the Application Landscape complexity?

18. Which types of costs can be linked with the above-mentioned factors, when considering a change to the landscape (e.g., project)?

Section 5 – Final Remarks

19. Do you think a simpler IT landscape can allow more agility in the adoption of future changes?

o Yes
o No

20. Can you describe the reasoning behind this choice?

21. Would a method for quantifying the Application Landscape complexity be valuable?

o Yes
o No

D.2 Complexity Drivers and Metrics

Informed Consent
Oral consent
– According to the Ethics Committee
- Participate voluntarily, Right to withdraw at any time
- No risks associated with the study
- Record the meeting

Introduction

The interview aims to gather additional information on the topic of IT application landscape complexity and to understand what requirements would be needed for the creation of a method to quantify complexity.

Facts about the interviewee

1. Have an introduction if it will be the first time meeting the interviewee (current role in company etc.)

Ways of Working

2. What are your daily tasks/routine?

3. Where does architecture (EA) play a role in your job?

Complexity

4. What does the complexity of an Application Landscape architecture mean to you?

5. Are you being hindered by complexity when performing any of your tasks? And if so which tasks/where do you see the effect of complexity?

6. Based on the outcome of the survey/ranking - look at extremes/similarities/differences etc.

a. Why do you consider that the level of complexity maturity capabilities was so dispersed? – show the chart and the levels; why did you score it like this; would you adjust your score?

b. Ask about the placement of factors in the ranking – Why did you order the factors in the survey ranking like this?

c. There was a trend showing that integrations, number of components and customised applications are the top complexity factors. Do you agree with this statement?

d. "Lack of documentation" was ranked the lowest, however, some people choose it as their first preference (it also appears to be lower for employees in higher positions). Why do you think that is the case?

e. What does customisation of an application mean to you?

f. Ask interviewee specific questions - .... ; *Here the link between costs and complexity can also be discussed*

7. How would you measure the factors: number of apps, the number of interfaces and the customisation based on the data currently available in your company?

Formula for quantifying complexity

8. From the key complexity factors we have discussed, what combination do you think could accurately indicate complexity?/how many would be sufficient to indicate complexity

9. How do you think these factors influence each other?

10. How do you think they influence the growth of complexity?

11. How would you use a formula for complexity? (calculate the complexity for one viewpoint, calculate the differences between an as-is and a to-be models etc.)

Method for applying the formula

12. How would you apply such a formula in your current way of working?

a. During the survey you mentioned A,B,C ... as possible tools or methods for quantifying complexity. How do you envision them to quantify the factor that we have just discussed?

b. What do you think about a script that can be used in Bizzdesign to quantify complexity?

i. If the interviewee has experience with scripting ask how should the process of running the script look like : e.g., go to Enterprise Studio – run the script – select a certain viewpoint etc.

ii. If the interviewee does not have experience with scripting, briefly introduce them to the concept

13. Show a scripting demo and ask if they have anything useful to add or to have based on it.

14. Would you require step-by-step/documentation on how to use such a method?

15. Follow-up interview when needed ...

D.3 Costs

Informed Consent
Oral consent
– According to the Ethics Committee
- Participate voluntarily, Right to withdraw at any time
- No risks associated with the study
- Record the meeting

Introduction

When talking about IT, complexity and costs, most of the costs associated with an IT landscape fall under the run and maintain costs. But what is the link between Application Landscape complexity and costs during change processes? The cost of complexity is believed to be proportional to the number of links and interactions of components.

This interview aims to uncover different types of costs that can be connected with the identified drivers of complexity.

Facts about the interviewee

1. Have an introduction if it will be the first time meeting the interviewee (current role in the company, daily tasks, where does EA play a role in their operations etc.)

Company Specific

2. What does complexity of an Application Landscape architecture mean to you? / Does it hinder any of your tasks?

Complexity

3. There was a trend in the survey answers showing that integrations, number of components and customised applications are the top complexity factors. Do you agree with this statement?

4. How would you measure these factors?

5. What combination indicates complexity for you?

6. How do the factors influence each other?

7. How do they influence the growth of complexity?

Costs

8. Is there a link between cost and complexity, from your perspective?

9. Are you familiar with cost taxonomies used in your company?

10. What are costs that are usually measured when undergoing a change in the Application Landscape?

11. Which costs can influence the decision-making process when selecting alternatives in a project? (e.g.; choose between different to-be scenarios)

12. Do you know which costs are estimated vs. which costs are known?

    a. How are these costs estimated?
    b. What do you consult when wanting to know more about costs?

13. During the survey, you indicated ... as costs - add more based on the answer (e.g., Why? How would you calculate the cost? From where do you get the data?)

14. The top three drivers of complexity mentioned in the survey are (I) the number of integrations/interfaces etc., (II) the number of components and (III) customised applications. Which costs do you expect to emerge during a change process for each?

Appendix E - Integration Style, Options and Middleware

In this appendix, the connection between the two disjoint integration styles mentioned during the case study, point-to-point (P2P) and reusable, is connected with the work of Hohpe and Wolf [141] on application integration options and middleware.

Two disjoint integration styles were identified during the case study. These are P2P and reusable integrations. P2P reflects the use of an interface specifically designed for the connection of only two applications, whereas a reusable integration focuses on an interface being consumed by multiple applications.

There are four types of application integration options:

1. File transfer - Each application produces files of shared data for the others to consume and each consumes what others have produced.
2. Shared database - The applications store what data they want to share in a common database.
3. Remote procedure calls (RPC) - Some of an application's procedures are exposed such that they can be invoked remotely. Other applications invoke those to run behaviour and exchange data.
4. Messaging - Each application connects to a messaging system to exchange data and invoke behaviour using messages.

Then, middleware is defined as the common standard communication mechanisms between applications [141].

Starting with the file transfer, if done in a P2P manner, this means that one application produces files specific to only another application. The latter consumes these files. However, to be done in a reusable style, this means that the files of shared data can be consumed by multiple applications.

Next, a shared database can mostly be associated with reusable integrations, as it represents a central place in which various applications store data and gather information. The database is standardised and managed via middleware, such that a common data layer is in place (e.g., middleware provides a consistent interface for all the applications). In the case of a P2P integration, the database access of an application is specifically tailored for another application.

With RPCs, in a P2P scenario, the mechanism is designed for a direct connection between two applications. For example, Application B calls specific functions in A. These functions are tailored to Application B's needs. If a change is made in Application A, then B must also change. In the context of a reusable integration, the RPC can be exposed through middleware which allows other applications to access the functions provided by Application A. For this particular example, Application A can evolve without impacting the consuming applications, as the middleware will handle the changes.

Lastly, message queuing can also be seen through both a P2P and a reusable integration lens. In a P2P scenario, the messages are set up for communication between two applications only. If changes emerge with the format or the configurations of the queue, both applications need to be updated accordingly. For reusability, however, the messages are managed via a message broker and multiple applications can access it. As in the case of

middleware for RPCs, here having the message broker allows one application to change independently of the others.

Based on the descriptions above, the following key takeaways need to be highlighted:

- P2P integrations - create tightly coupled systems in which one change in an application affects the other; specific file transfer and message queues or direct database queries or RPC calls between two applications are examples of these P2P integrations.
- Reusable integrations - as opposed to P2P these support a loosely coupled architecture and standardisation via middleware or message brokers; standardised file transfers, RPC calls, common databases and a central message broker can be considered as part of the reusable integration category.

Appendix F - Scripts in ArchiMate

F.1 Counting Number of Interfaces

```
viewOfPlateau="";
objectReferenceToView="";
numberOfFlows=0;

forall "IMPlateau" p in modelpackage{

   viewOfPlateau = p.attr("interlink").toString().split(",")[2];

   /*Checks if Plateau is assigned to a view and remembers the view */
   forall "MM_Object" mm in modelpackage {
      check=" "+mm.toString();
      if (check==viewOfPlateau){
         objectReferenceToView = mm;
      }
   }

   /*In the view associated with a Plateau calculate the amount of flows*/
   numberOfFlows=0;
   forall elem in objectReferenceToView {

      if (elem.type().toString()=="Flow relation")
         numberOfFlows = numberOfFlows + 1;
   }

    result = Structure();
    result.add("object", p);
    result.add("value", numberOfFlows);
    output result;
}
```

F.2 Counting Application Functionality Overlaps

```
viewOfPlateau="";
objectReferenceToView="";

numberOfFunctionalityOverlaps=0;

forall "IMPlateau" p in modelpackage{

   viewOfPlateau = p.attr("interlink").toString().split(",")[2];

   /*Checks if Plateau is assigned to a view and remembers the view */
   forall "MM_Object" mm in modelpackage {
      check=" "+mm.toString();
      if (check==viewOfPlateau){
         objectReferenceToView = mm;
      }
   }

   /*In the view associated with a Plateau calculate how many apps realize
       services*/
        indexFunctionality = Index ();
        forall "ApplicationService" aS in model {

              /*Check if the service in the model is part of the view
              If it is then perform a certain action
              */
              forall elem in objectReferenceToView {
                 if (elem.type().toString()== "Application service"
                       && aS.toString()== elem.toString()){
                            /*Count how many applications are realising a
                                service*/

                            realisedByApplications = aS.objectsWithRole( "is
                                realised by", "ApplicationComponent");

                            /*Loop through all the applications which are
                                associated to a data object and see which ones
                                are present in a view*/
                            n=1;
                            counter=0;

                                    while(n<=realisedByApplications.size()){

                                       /*Check if that application is part of
                                           that view and count*/
                                       forall acInView in objectReferenceToView {
                                          if(acInView.type().toString()=="Application
                                              component" &&
                                          acInView.toString()==
                                              realisedByApplications[n].toString()){
                                             counter=counter+1;
                                          }
                                       }
                                    n=n+1;
                                 }
```

```
                        if (counter>1){
                        /*If a service is realised by multiple applications,
                            mark the ones doing the same thing.
                        Use index to make sure that the app is counted only
                            once*/
                           forall elem in realisedByApplications {
                                   indexFunctionality.add(elem, 1);
                             }
                        }
                    }
                }
            }
            /*Get the number of the index size*/
            numberOfFunctionalityOverlaps = indexFunctionality.size();

    result = Structure();
       result.add("object", p);
       result.add("value",numberOfFunctionalityOverlaps);
    output result;

}
```

F.3 Counting Data Overlaps

```
viewOfPlateau="";
objectReferenceToView="";

numberOfDataOverlaps=0;

forall "IMPlateau" p in modelpackage{

   viewOfPlateau = p.attr("interlink").toString().split(",")[2];

   /*Checks if Plateau is assigned to a view and remembers the view */
   forall "MM_Object" mm in modelpackage {
      check=" "+mm.toString();
      if (check==viewOfPlateau){
         objectReferenceToView = mm;
      }
   }

   /*In the view associated with a Plateau calculate how many apps are using the
       same business object.*/
        indexData = Index ();
        forall "ApplicationDataObject" dO in model {

              /*Check if the object in the model is part of the view. If it is
                  then perform a certain action
              */
              forall elem in objectReferenceToView {
                 if (elem.type().toString()== "Data object"
                       && dO.toString()== elem.toString()) {
                            /*Count how many applications are associated with a
                                business object*/

                            associatedWithApplications = dO.objectsWithRole(
                                "associated with", "ApplicationComponent");


                            /*Loop through all the applications which are
                                associated to a data object and see which ones
                                are present in a view*/
                            n=1;
                            counter=0;

                                    while(n<=associatedWithApplications.size()){

                                       /*Check if that application is part of
                                           that view and count*/
                                       forall acInView in objectReferenceToView{
                                          if(acInView.type().toString()=="Application
                                              component" &&
                                          acInView.toString()==
                                              associatedWithApplications[n].toString()){
                                             counter=counter+1;
                                          }
                                       }
                                    n=n+1;
```

```
                                    }

                                if (counter>1){
                                /*If a business object is used by multiple
                                    applications, mark the ones doing the same thing.
                                Use index to make sure that the app is counted only
                                    once*/
                                   forall elem in associatedWithApplications {
                                     indexData.add(elem, 1);
                                     }
                                }
                        }
                    }
                }
                /*Get the number of the index size*/
                numberOfDataOverlaps = indexData.size();


    result = Structure();
       result.add("object", p);
       result.add("value",numberOfDataOverlaps);
    output result;
}
```