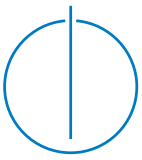
FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Information Systems

Tool Support for Capability-Based Application Portfolio Management - Conceptualization, Prototype Implementation, and Evaluation

Fatih Yılmaz



FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Information Systems

Tool Support for Capability-Based Application Portfolio Management - Conceptualization, Prototype Implementation, and Evaluation

Werkzeugunterstützung für das Applikationsportfolio-Management mit Hilfe von Business Capabilities

Author:	Fatih Yılmaz
Supervisor:	Prof. Dr. Florian Matthes Fakultät für Informatik Technische Universität München
Advisor:	Pouya Aleatrati Khosroshahi, M. Sc. Fakultät für Informatik Technische Universität München
Date:	January 11, 2017

I assure the single handed composition of this master's thesis is only supported by declared resources.

Munich, January 11, 2017 Fatih Yılmaz

Abstract

Today, increasing complexity of company's application portfolio became a major challenge for enterprise architects. Over time changing business requirements lead to a historically grown application portfolio, which is difficult to manage, cause high costs and cannot be adapted easily to meet new requirements. Consequently, Application Portfolio Management (APM) was introduced as an approach to manage applications from a holistic point of view. Nevertheless, it is difficult to steer a portfolio containing hundreds or thousands of applications. Business Capability Modeling could be an approach to address this problem. Business capabilities provide insights into the company's competences which are necessary to process value creating activities and create a holistic view of the organization's main functions. This view is captured in a business capability map. Beside providing an overview, it can also be used for strategic planning of IT. By mapping underlying IT components to business capabilities, the status of the application architecture can be visualized and better managed. Therefore, it attracts increasing attention in the Enterprise Architecture Management (EAM) community over the last years.

Capability-based APM primarily describes the combination of APM and Business Capability Modeling. By means of this approach, this thesis presents Key Performance Indicators (KPIs) to evaluate the application architecture of business capabilities. Currently, KPIs are used to evaluate the application architecture in many different ways. But mostly they provide very limited information, by measuring only a few application specific attributes. Hence, one goal of this thesis was to develop higher-level KPIs in order to provide a more general view. By conducting review on relevant literature and interviews with experts, three KPIs are developed, which evaluate the application architecture in terms of complexity, quality and failure impact. Since companies have recognized the benefits of Business Capability Modeling, many of them have implemented a capability map. Therefore, the second goal of the thesis was to evaluate the use of the business capability map in order to visualize the application architecture's status. For this purpose, a prototype is presented based on a suitable design which is identified by literature review and requirements analysis.

Contents

Abstract **vii**

Outline of the Thesis **xi**

1. Introduction **1**
1.1. Motivation . . . 1
1.2. Research Questions . . . 3
1.3. Research Approach . . . 4

2. Foundations **7**
2.1. Enterprise Architecture . . . 7
2.2. Enterprise Architecture Management . . . 9
2.2.1. Zachman Framework . . . 10
2.2.2. TOGAF . . . 11
2.3. Application Portfolio Management . . . 13
2.4. Business Capability Modeling . . . 14

3. Related Work **19**
3.1. Literature Review Approach . . . 19
3.2. Findings of Literature Review . . . 21
3.3. Limitations of Related Work . . . 22

4. KPI Conceptualization for Application Architecture Evaluation **25**
4.1. Application Portfolio Rationalization Methods . . . 25
4.2. Application Characteristics . . . 29
4.3. Application Landscape Evaluation . . . 35
4.3.1. KPI Requirements . . . 35
4.3.2. Approach for KPI Development . . . 37
4.3.3. KPI Categorization . . . 43
4.3.4. KPI for Measuring the Application Architecture Complexity . . . 45
4.3.5. KPI for Measuring the Application Architecture Quality . . . 49
4.3.6. KPI for Measuring the Impact of Application Architecture Failure 51
4.4. Recommendations for Action . . . 53
4.4.1. Reduction of Application Architecture Complexity . . . 53
4.4.2. Increase in Application Architecture Quality . . . 54
4.4.3. Avoid serious consequences of Application Landscape Failure . . 55

5. Case Study **57**
5.1. Data Gathering 57
5.1.1. Data Collection and Cleansing 57
5.1.2. Multicollinearity Test 59
5.2. Application Architecture Evaluation 59
5.3. Recommendations for Action 61

6. Application Architecture Status Visualization **63**
6.1. Data Visualization Types in Enterprise Architecture 64
6.2. Prototype Development 65
6.2.1. Requirements Analysis 65
6.2.2. Prototypical Design 73
6.2.3. Data Model 75
6.2.4. Prototypical Implementation 76

7. Evaluation of the Artifacts at a European Automotive Company **79**
7.1. Evaluation Approach 79
7.2. Evaluation Results 79

8. Conclusion **83**
8.1. Summary and Conclusion 83
8.2. Limitation 84
8.3. Future Work 85

Appendix **89**

A. Appendix **89**
A.1. Application Characteristics Correlations 89
A.2. Prototype Screenshots 90
A.3. Evaluation Questionnaire 94
A.4. Evaluation Results 98

Bibliography **101**

Outline of the Thesis

CHAPTER 1: INTRODUCTION
This chapter describes the problem in hand by revealing a motivation for status evaluation of application architecture by means of business capabilities. Furthermore, the derived research questions and the research approach will be discussed.

CHAPTER 2: FOUNDATIONS
To provide a unified understanding the theoretical foundations of the thesis are explained. Thus, the scope of EA, EAM, APM and Business Capability Modeling will be defined.

CHAPTER 3: RELATED WORK
In this part the key findings on related work, including relevant literature on application architecture evaluation and the existing options for status visualization are discussed. Additionally, it unveils their limitations.

CHAPTER 4: KPI CONCEPTUALIZATION FOR APPLICATION ARCHITECTURE EVALUATION
This chapter describes the process of application architecture evaluation by KPIs. Thus, it includes the identification, adaption and application of several KPI development frameworks. Furthermore, recommendation for actions are presented in order the improve the application architecture's status.

CHAPTER 5: CASE STUDY
In order to evaluate the developed KPIs, they are applied on a data set given by a European automobile manufacturer in form of a case study. After assessing their application architecture, a recommendation for action is proposed.

CHAPTER 6: APPLICATION ARCHITECTURE STATUS VISUALIZATION
To illustrate the application architecture status, the outcome of the KPIs are mapped on business capabilities. Therefore, existing visualization methods in EA are analyzed and a data model created. Based on this information, a prototype is implemented.

CHAPTER 7: EVALUATION OF THE ARTIFACTS AT A EUROPEAN AUTOMOTIVE COMPANY
In this chapter the need for application architecture status monitoring, the outcome of the developed KPIs and the prototype are evaluated by experts from the industrial partner. The results of the interviews are captured in a questionnaire.

CHAPTER 8: CONCLUSION
Here the results of the thesis are summarized and its limitations shown. Additionally, based on the insights and feedback gathered during the thesis, recommendations for

future work are made.

1. Introduction

This chapter provides the motivation for the thesis in section 1.1 and describes its objectives with corresponding research questions in section 1.2. Afterwards, the underlying research approach in order to answer these questions is presented in section 1.3.

1.1. Motivation

"You can't manage what you can't measure" and "What gets measured gets done" are both decades-old sayings which emphasize the importance of measurement. Performance measurement is essential to all organizations around the world, whether they are multi-million dollar enterprises or small charities. It helps organizations to align daily activities to strategic objectives in order to ensure their coherence [Par11]. Performance measurement is also applied in the field of Information Technology (IT). Especially in APM which is concerned with the conduction of application portfolios. Today's organization's applications portfolio comprises hundreds or even thousands of applications. Thus, the application architecture represents a central part of every enterprise's IT infrastructure. Besides the business applications, the application architecture also covers typical IT architecture elements such as information flows, technological components and platforms. Furthermore, it includes the people developing, operating and managing those elements [Sch16]. This complex nesting of different components and involved stakeholders makes the application architecture to a very rigid construct. However, business applications enable the operative business and are thus crucial for the business success. Yet business requirements can change frequently which issues a challenge for the existing application architecture and the underlying IT infrastructure. Considering that, the enterprise's IT architecture elements should be easily adaptable and thus show high level of agility in order to meet the changing business requirements. But resulting from the application architecture's complexity and its rigidness many organizations achieve this by adding new applications and IT components into their portfolio. This workaround leads to more information flows between applications, thus higher application architecture complexity. Consequently, the adaption of the existing systems for further business requirements is getting even more difficult and forces the organization to spend huge amounts of money in order to maintain their systems.
One of the main objectives of APM is the reduction of application portfolio complexity [SFS10]. Therefore, several scientists from the IT domain already have conducted research on this topic [AKBA16, Moc09]. Yet an investigation in terms of quantitative performance measurement is missing. This can be achieved by use of KPIs. In or-

der to monitor and measure the status of an organization's infrastructure, KPIs have been established as an appropriate tool. Speaking ideally, it quantifies the status and thus the organization's achievements in a unbiased way. There are many reasons why KPIs are increasingly used for performance measurement. Despite that, most managers are struggling to interpret and identify the right KPIs companies are collecting a vast number of figures that are easy to measure [Mar12]. Hence the organization expends a tremendous amount of time and effort ascertaining KPIs without value added. The following statement made in a German insurance company boils the problem down to an essence:

> *"We had perhaps more than 80 of them [KPIs] throughout the business. Our objective was to achieve a increase in KPI visibility across our group, and understand what drives us."* [GH13]

Summarized, the number of applications within an application portfolio is increasing over time and for each application an enormous number of measures exists. This leads to a lack in transparency and missing interpretation possibility. Consequently, the KPIs do not support the decision-making process at all. Therefore, the well-known KPI scientist David Parmenter advices to implement at most ten organization-wide KPIs [Par11]. Inspired by this statement, the thesis aims to develop high-level KPIs which evaluate the overall status of the application architecture. A special focus will be on complexity measurement since this topic has attracted a lot of attention in the APM community.
During expert interviews the illustration of KPIs has been crystallized as an additional problem. Most of the EAM tools still lack in clear structured, but customizable visualizations [Rot14]. Therefore, many companies use these tools only as a repository but can rarely provide useful illustrations to support the decision-making process. This problem will be addressed by developing a prototype design based on suitable visualization types in EAM. Since Business Capability Modeling became an emerging topic in the EAM, the business capability map represents the foundation of the visualization design. This approach further shows the relationship between APM and Business Capability Modeling which is neglected by literature so far.

1.2. Research Questions

This thesis aims the conceptualization of KPIs and tool design for a capability-based evaluation of the application architecture. It has to be noted that the research questions are defined during discussions with a European automobile manufacturer. Based on challenges they face and lack in scientific literature the following three questions are deducted:

- **Research Question 1:**
  What kind of application characteristics can be used to evaluate the status of application architecture?
- **Research Question 2:**
  How can the business capability map be used to visualize the application architecture's status?
- **Research Question 3:**
  What kind of operational actions can be derived from the application architecture's status?

To answer these questions appropriately the accepted research design methodology *literature review* is conducted [Bak00, Yin13]. When undertaking a research project, literature review is one of the most essential parts. Usually it is done at the first place to uncover related topics and publications.
The research questions are designed in such a way that they build upon each other. Each research question provides insights or results which contributes in answering the next research question. Their interconnectedness and the applied methodology is illustrated in figure 1.1. The first research question identifies appropriate characteristics which can help to evaluate the application architecture. Therefore, existing literature regarding general application attributes is reviewed. In this context, the thesis is particularly focusing on quantitative attributes which can be numerical data or data that can be transformed into distinct statistics. The second part of this research question is the conceptualization of KPIs. Based on the various application characteristics identified in literature, the status of the application architecture is evaluated. For this purpose, high-level KPIs will be developed. The accuracy and correctness of the KPIs is ensured by iterative attempts of expert discussions.
The second research question is based on the results of the first one. It targets the visualization of the application architecture status. Since this thesis focuses, among others, on Business Capability Modeling, a business capability map will form the foundation of the visualization. In addition, as part of proof of concept a prototype will be presented. This research question comprises a literature review on existing visualization types to design the prototype.
The last research question addresses the consequences which can derived from the status of the application architecture. Based on the visualization, hot spots in the application architecture are identified and recommendations for status improvement of poor

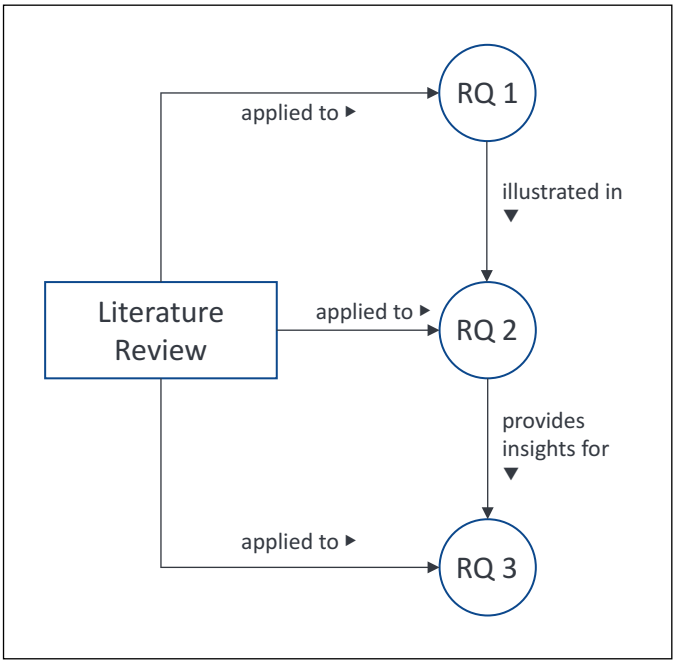


Figure 1.1.: Research question connectedness

applications will be presented. Therefore, another literature review in the field of APM will be conducted.

1.3. Research Approach

To answer the previously described research questions in a scientifically accurate way the structure of this thesis refers to the *design science* paradigm presented by Peffers [PTRC07] which is based on Hevner's prominent *Design Science in Information Systems Research* [VAMPR04]. Both paradigms represent design science research methodology approaches applied towards the information systems research area. Originally this approach consists of six consecutive steps performing iteratively actions. For the purpose of this thesis the fourth and fifth steps are merged together since the *demonstration* and *evaluation* part are performed simultaneously. The adapted design science approach with the resulting artifacts and the related chapters is illustrated in figure 1.2.

In the first step the problem is defined and its importance pointed out. For this purpose, Peffers suggests a problem-centered initiation. In the presented thesis, this point is conducted in the first chapter. To summarize, the existing problem is the missing holistic view of the application architecture's health status in combination of business capabilities. Constructively, there is a lack of research in linkage between APM and Business Capability Modeling. For conducting the second step Peffers asks the question: *What would a better artifact accomplish?* Here the objectives of the solution are de-

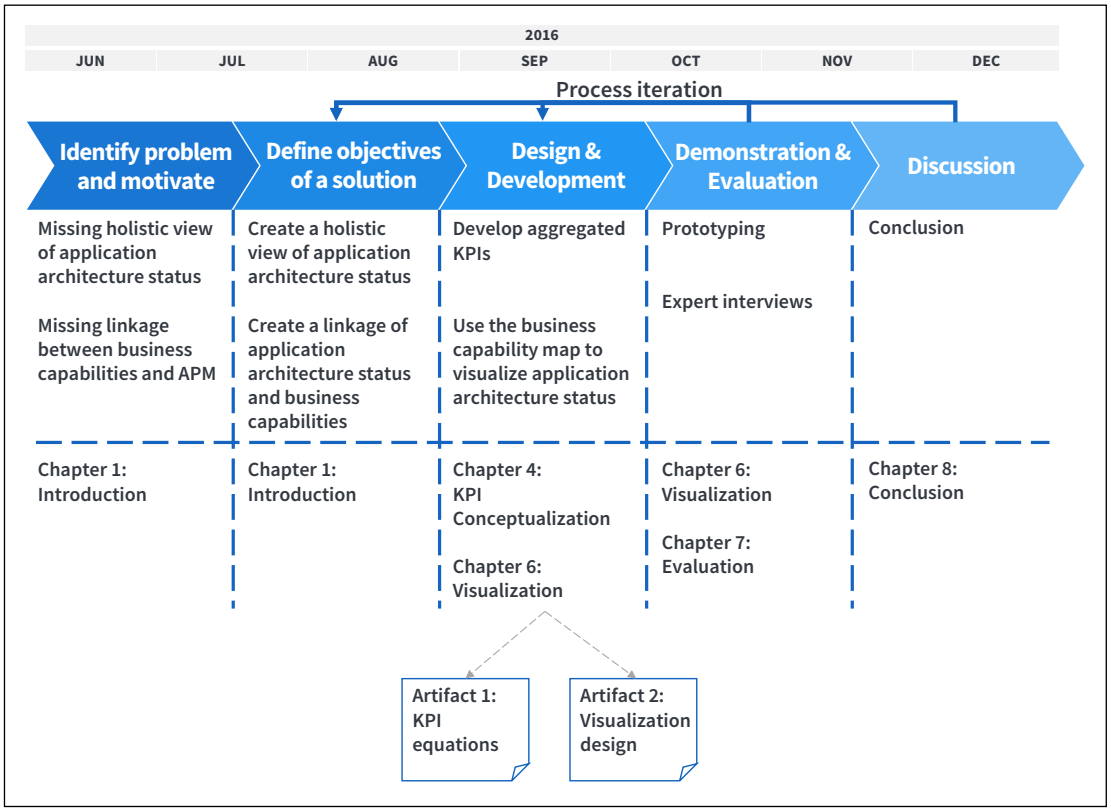


Figure 1.2.: Overview of the underlying research approach according to Peffers [PTRC07] and Hevner [VAMPR04]

ducted from the problem definition which took place in the previous step. Generally speaking, this thesis comprises two objectives. In order to solve the problem a holistic view of the application architecture status has to be created. Additionally, a method has to be presented, which allows the creation of an appropriate linkage of the application portfolio and business capabilities. During the *design & development* phase the actual artifacts are created. The concrete artifacts in this thesis are the equations to calculate the KPIs and the visualization design to illustrates the application architecture's status. It should be noted that this thesis is of a conceptual nature. The KPIs and the visualization are intended to show a proof of concept. Therefore, the technical implementation of the prototype was conducted with firm support of the Software Engineering for Business Information Systems (sebis) chair of the Technische Universität München. This prototype is applied on a case study with data provided from the industrial partner. The results and the corresponding process of data collection and cleansing is described in chapter five. The creation of the artifacts is presented in chapter four and six. The artifacts are evaluated by expert interviews in chapter seven. Since Peffers' design science includes iterative processes, the created KPIs and the visualization are revised multiple times. For that reason, biweekly jour fixes with enterprise architects are conducted.

The final *discussion* step shortly summarizes the problem, the presented artifacts and the evaluation results. Further, limitations are shown and future works proposed. This phase is described in chapter 8.

2. Foundations

This chapter comprises an overview of fundamental terminologies, which are relevant for this thesis. The purpose of this chapter is to establish a unified understanding of the related IT disciplines and to classify this thesis within these disciplines. Therefore, in the following sections the terms *EA*, *EAM*, *APM* and *Business Capability Modelling* will be defined.

2.1. Enterprise Architecture

Generally, *architecture* is defined as *"the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution"* [C+07]. In context of IT, *The Open Group Architecture Framework (TOGAF)* describes architecture in two different perspectives. Firstly, architecture is *"a formal description of a system, or a detailed plan of the system at component level to guide its implementation"* and secondly, architecture is *"the structure of components, their inter-relationships, and the principles and guidelines governing their design and evolution over time"* [Ver09]. Since EA evolved to a popular research topic during the last couple of years, there is a variety of definitions for it [C+07, Sch08, KSS15, RWR06]. But put simply, EA outlines a blueprint of the company's *IS* [Mas06]. For a unified understanding in this thesis EA is defined in accordance with the ISO Standard 42010 as

> *"the fundamental conception of the organization in its environment, embodied in its elements, their relationships to each other and to its environment, and the principles guiding its design and evolution."* [C+07]

One of the goals of EA is to optimize the legacy of processes into an integrated environment. This provides a strategic context for the adaption of IS in response to the constantly changing needs of the business environment [Ver09]. For that reason, EA defines a holistic view of the enterprise IT architecture instead of taking view on single applications [KAV05]. Nevertheless, the management of applications is an important part of EA (see section 2.3). Although documentation of the architecture in layers has proven to be useful, there are different layer models in literature [WF06, Ver09]. To ensure unambiguity this thesis refers to the approach presented by Buckl (2011) which consists of three layers [Buc11].

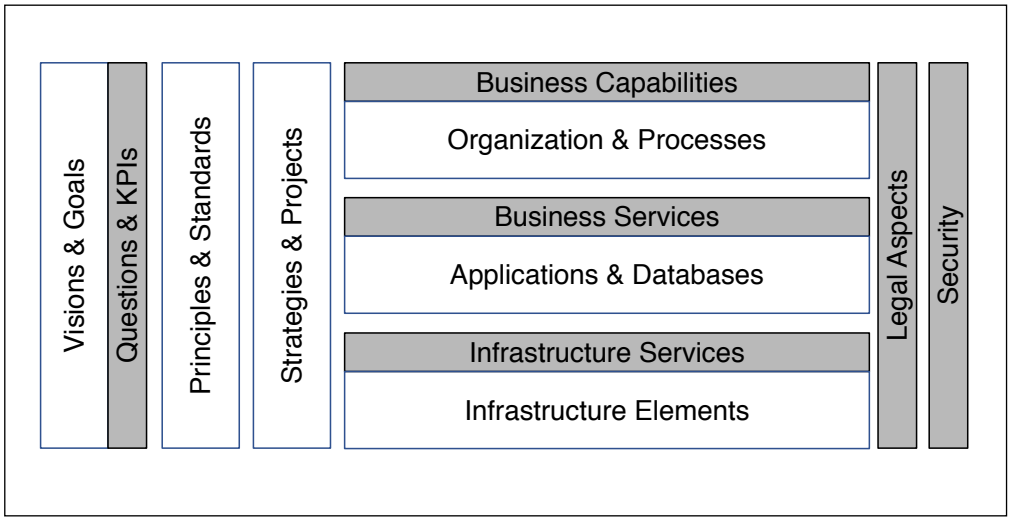


Figure 2.1.: Layered approach to EA documentation according to Buckl (2011) [Buc11]

This model includes the *business & organization* layer, the *application & information* layer and the *infrastructure & data* layer. The business & organization layer describes the organization related aspects of the enterprise. The application & information architecture layer, where special attention is devoted to in this thesis, focuses on the business applications and their interfaces to each other. At the end, the infrastructure & data layer provides the foundation for the previous layers. It contains the technical infrastructure components, such as hardware devices. Further, additional abstractions like *business capabilities*, *business services*, *infrastructure services* or *questions & KPIs* encapsulate the architecture layers. At the top, the business capabilities describe the activities of the enterprise which play a part in creating value. The abstraction layer business services comprises the company internal business services and processes which enable the business capabilities. Technical services enabling the operation of the infrastructure & data layer are provided by the infrastructure services abstraction layer. Questions & KPIs support the cross-cutting element *visions & goals* in order to achieve them. Those cross-cutting elements which are illustrated vertically are not part of, but have influence on any of the layers. It is also important to note that each layer interacts with other elements in the model. This reflects the business IT alignment, which is denoted as *"central to the IS discipline"* by Hevner (2004) [VAMPR04].
Since one objective of this theses is to evaluate the application architecture in context of business capabilities, the above-mentioned abstractions business capabilities, question & key performance and the layer application & information play a central role. Since there is no clear distinction in terminology, the application architecture is defined according to Braun's understanding of application landscape. In his opinion the application landscape is a part of the EA and forms an open system which is influenced by the enterprise. Compared to the application & information layer presented by Buckl (2011), describes Braun (2005) the application landscape as a set of applications and

their interdependencies [BW05]. Furthermore, it is derived by analyzing similarities regarding data access, ownership, process support and functional reuse.

2.2. Enterprise Architecture Management

Roger Sessions from ObjectWatch, Inc. compares an enterprise with a big city. He says that building a large, complex enterprise-wide IS without an enterprise architect is like trying to build a city without a city planner [Ses07]. Therefore, the role of enterprise architects is crucial for a successful and continual IS. The activities which have to performed in order to establish and develop an EA are structured in the IT Governance function of EAM [AGW11]. Exemplary EAM activities are:

- Defining the IT strategy
- Modeling architectures
- Evolving the IT landscape
- Assessing and developing capabilities
- Developing and enforcing standards and guidelines
- Monitoring the project portfolio
- Leading or coaching projects
- Managing risks involved in IT [BBL12]

When discussing EAM, many different definitions come into light [MBLS08, WF06, Lan09, RWR06]. Since no general definition of the term EAM exists, the following holistic definition is used in this thesis:

> *"Enterprise Architecture Management is a continuous and iterative process controlling and improving the existing and planned IT support for an organization. The process not only considers the IT of the enterprise, also business processes, business goals, strategies etc. are considered in order to build a holistic and integrated view on the enterprise. Goal is a common vision regarding the status quo of business and IT as well as of opportunities and problems arising from these fields, used as a basis for a continually aligned steering of IT and business."* [MBLS08]

EAM is often performed in distributed teams assigned with a multitude of tasks. Enterprise architects continuously have to collect, store, aggregate, analyze and visualize data on current, planned and target EA states. Thus, many EAM tools have been developed over the last years. Since each tool has different strengths and weaknesses, the sebis chair of the Technische Universität München has evaluated a variety of the prominent EAM tools [MBLS08].

An Enterprise Architecture Framework (EAF) is *"a skeletal structure that defines suggested architectural artifacts, describes how those artifacts are related to each other, and provides generic definitions for what those artifacts might look like"* [Ses07]. Based on *"a set of assumptions, concepts, values and practices"* [BBL12] a fundamental structure of the EA can be created. In detail, many EAFs provide a method which describes the end state of the enterprise in terms of a set of building blocks and the relationships between them. According to Sessions, EAFs should consist of a description and a method by which the deliverables should be produced, which they mostly don't do [Ses07]. Over the last years, a tremendous amount of different EAFs have been developed. To give an overview of existing EAFs, Matthes has created a comprehensive list containing 50 EAFs [Mat11].
In the following the two popular EAFs *Zachman Framework* and *TOGAF* will be presented.

2.2.1. Zachman Framework

It is almost 30 years ago, when J. A. Zachman draw attention to the topic of EA for the first time. With his publication in the IBM Systems Journal of an article titled *A Framework for Information Systems Architecture*, he pointed to the challenge of managing the complexity of increasing distributed systems:

> *"The cost involved and the success of the business depending increasingly on its information systems require a disciplined approach to the management of those systems."* [Zac87]

The initial version of the *Zachman Framework* contained five levels and three perspectives. Later in 1992 Zachman extended the framework by the additional three perspectives *Persons (Who?)*, *Time (When?)* and *Motivation (Why?)*. Furthermore, a meta-model for the owner's, designer's and builder's level was added [SZ92]. The latest version of the framework is shown in the figure 2.2.
Zachman defined different levels and perspectives in order to create a logical structure. Each row represents a view of the solution from a particular perspective. Similar to the layered approach of EA presented by Buckl (2011), the rows are hierarchically structured. That means, the lower rows affect the rows above. But higher rows do not necessarily affect the rows below [Cou99]. A more detailed description of the framework will not be given here, since it would go beyond the scope of the thesis. Going by the thesis objective of application architecture evaluation, the relevant part of the framework is the intersection of designer's view (third row) and the functional perspective (second column). It is difficult to order *Business Capability Modeling* into the Zachman Framework. As already noticed by Nick Malik, *"a business capability is an architectural concept that does not exist in the Zachman framework"* [Mal09]. He says, that business capabilities are composed of multiple elements of the Zachman Framework. Botha & Tshwane endorse this statement by suggesting that *"all cells of the Zachman Framework could be used collectively to compile business capabilities"* [BBH07].

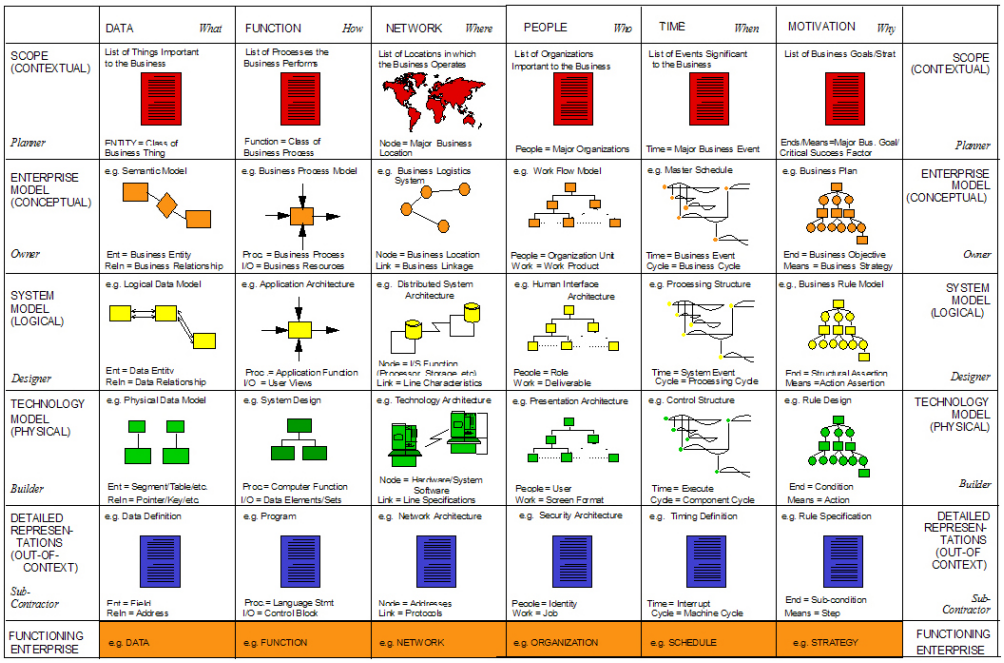


Figure 2.2.: Zachman Framework for Enterprise Architecture, 1992 [SZ92]

2.2.2. TOGAF

The *Open Group* architecture forum that develops IT standards, has published *The Open Group Architecture Framework* in 1995. Since then it has been improved continuously. Currently, the latest TOGAF version is 9.1. In addition to describing building blocks and their relationships, TOGAF also provides a set of tools, a common vocabulary, a list of recommended standards, and compliant products that can be used in implementing the building blocks [Lum16]. TOGAF consists of the following seven parts [Ver09]:

- **Part I: Introduction**
  This part provides a high-level introduction to the core concepts of EA and the TOGAF approach in particular. It contains the terminologies used in TOGAF and notes describing the changes between releases.

- **Part II: Architecture Development Method (ADM)**
  The TOGAF Architecture Development Method (ADM) is the core of TOGAF. It describes how to develop an EA in a phase-oriented approach.

- **Part III: ADM Guidelines and Techniques**
  This section includes a set of methods and guidelines available supporting the use of TOGAF and the TOGAF ADM.

- **Part IV: Architecture Content Framework**
  This part describes the TOGAF content framework. It presents a structured meta-

model for architectural artifacts, re-usable architecture building blocks, and a catalog of typical architecture deliverables.

- **Part V: Enterprise Continuum and Tools**
  This part discusses taxonomies and tools within TOGAF which stores and categorizes the enterprise architecting results of the organization.

- **Part VI: Reference Models**
  The Reference Models section, e.g., the TOGAF Technical Reference Model (TRM), provides a collection of architectural reference models, which includes the TOGAF Foundation Architecture, and the Integrated Information Infrastructure Reference Model (III-RM).

- **Part VII: Architecture Capability Framework**
  This section discusses the necessary organization, processes, skills, roles, and responsibilities which are needed to establish and operate an architecture function within the organization.

The figure 2.3 illustrates the TOGAF elements and their relationship to each other. As mentioned previously, this thesis focuses on the application architecture and business capabilities of the enterprise. Just like in the Zachman Framework the application architecture is a major part of TOGAF. Being part of the TOGAF ADM, *"the objective [of the application architecture] is to define the major kinds of application system necessary to process the data and support the business"* [Ver09]. For the evaluation of application architecture also important is part VII: *Architecture Capability Framework*. By setting targets, KPIs and plans it is in close contact with the TOGAF ADM. Compared to the Zachman Framework business capabilities are considered by the framework (see figure 2.3).
Both, the Zachman Framework and TOGAF are adding value to the company's EA. Zachman provides a simple overview of the architecture while TOGAF gives a more detailed view of multiple aspects of EA. In sum, TOGAF is a more comprehensive and contemporary EAF and thus become very popular in the last few years. Nonetheless, each EAF has his strengths and weaknesses. Depending on the case in hand one EAF can be more suitable than another. One they have in common, they all support enterprise architects in developing better EA.

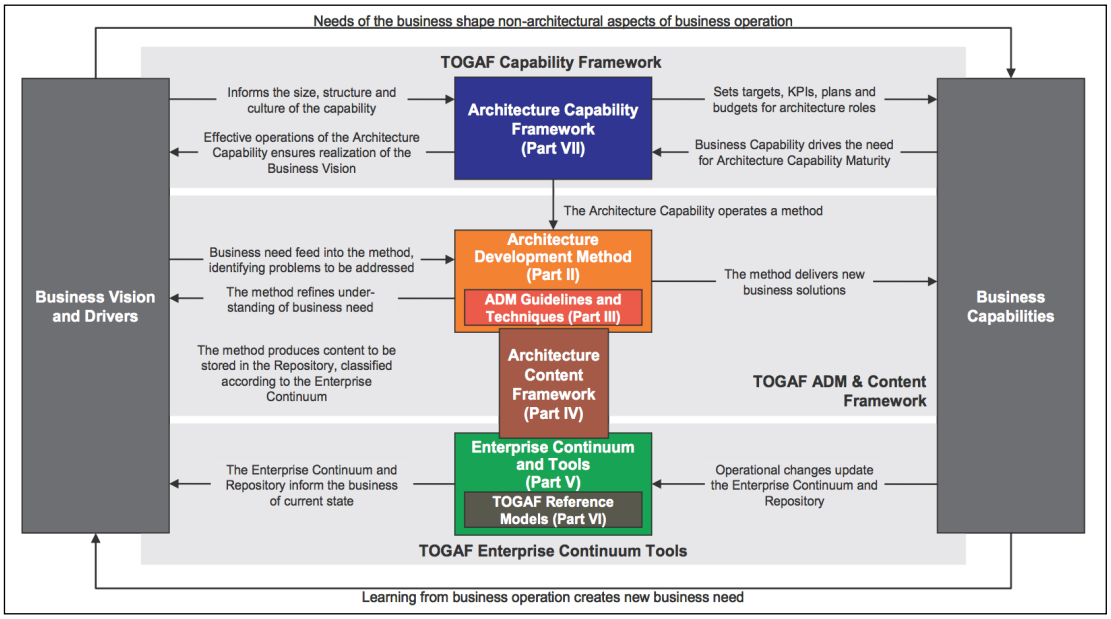


Figure 2.3.: Structure of the TOGAF document [Ver09]

2.3. Application Portfolio Management

The terminology of *portfolio* was introduced by H.M. Markowitz back in 1952. He presented the Modern Portfolio Theory (MPT) as an approach to minimize risk of investments. The aim is to create a so-called pareto-optimal portfolio by combining financial assets (e.g., stocks) of different fields [Mar52]. About 20 years ago the IT management expert Cyrus Gibson published the earliest and most famous application portfolio model [GN74]. Together with Richard Nolan he presented a study on managing application growth in four different stages. In the first stage the company uses applications only for accounting. Than applications are implemented in all functional areas (e.g., sales). In stage three the focus is on managing the existing applications. Back in 1974, the implementation of data-base applications represented the last stage of application growth. To manage each stage properly, for each stage of application growth an activity regarding IT-personnel and management techniques are presented [GN74]. Ever since the research field of APM become increasingly popular. In literature, there are usually two different perspectives on APM. One is a very finance biased view on applications inspired by Markowitz's MPT. The focus here is on calculating financial metrics (e.g., Return on Investment (ROI) or Business value of IT (ITBV) [Swa06]) in order to assess the cost effectiveness of business applications [BAC11]. The other perspective on APM allows a more general and EA-oriented view on the application architecture [RGA07]. Beside cost aspects, in this view business process and capability support, business objectives and missions, underlying technologies and other environments are taken into account [Wal07]. Generally speaking, an application portfolio describes all the appli-

cations run by a specific enterprise [RGA07]. Consequently, APM is the process of managing a set of applications or the entire application architecture. A contemporary definition of APM is given by Simon (2010):

> *"Application Portfolio Management is the ongoing application of systematic and structured decision-making processes to evaluate an organization's applications along various dimensions (from a business and a technical viewpoint), weigh various actions for the purpose of optimization, and implement appropriate actions to resolve identified issues and meet key enterprise objectives. The promise of Application Portfolio Management lies primarily in reducing the complexity of the application landscape, which is approached from a holistic viewpoint."* [SFS10]

Drawing on the definition above, Simon (2010) developed a comprehensive APM framework which includes the APM process, a view on the APM maturity and other APM-related aspects. In scientific literature, a variety of APM processes exist [WV99, Sar06, FBvD07, Kro09, SFS10, GBB12]. Yet all the mentioned processes can roughly be divided into the three phases: situation assessment, assessment evaluation and plan actions [FBvD07]. Since, two of this thesis' research questions cover this topic, it will be further discussed in chapter 4.
Maizlish & Handler (2005) defined four goals of APM. Besides providing a basis for application-related discussions and the identification of business strategy consistent applications, APM seeks to communicate the status of the existing application set in order to identify major issues associated with each application. The last-mentioned goal of APM is closely related to the objectives of this thesis.

2.4. Business Capability Modeling

A multitude of different *capability* definitions can be found in scientific literature. In many cases the intended meaning depends on the context which the term is being used. One of the most cited definitions is presented by the US Department of Defense (DoD). They describe capability as *"the ability to achieve a desired effect under specified standards and conditions through combinations of means and ways to perform a set of tasks"* [oD09]. Although this definition precisely describes what a capability is, a IT-related definition would be more concrete. For that matter, this thesis refers to the capability definition presented by TOGAF:

> *"Capability is an ability that an organization, person or system possesses. Capabilities are typically expressed in general and high-level terms and typically require a combination of organization, people, processes, and technology to achieve. For example, marketing, customer contact, or outbound telemarketing"* [Ver09]

Business capability is the combination of the term capability and organizational value creation [Ros10, CK09, UR11, Gre09]. It is *an ability of capacity for a company to deliver value, either to customers or shareholders.* [Gre09]. The process of creating business capabilities is described as Business Capability Modeling. Figure 2.4 represents the hype

cycle for EA that demonstrates the increasing importance of Business Capability Modeling. According to Gartner Inc. it is positioned as a peak of inflated expectations. This stage of the hype cycle is defined as *early publicity which produces a number of success stories - often accompanied by scores of failures. Some companies take action; many do not* [Gar15]. It is expected to reach the plateau of productivity in two to five years, where the mainstream adoption takes place. The Cutter Consortium's Executive Report *The*

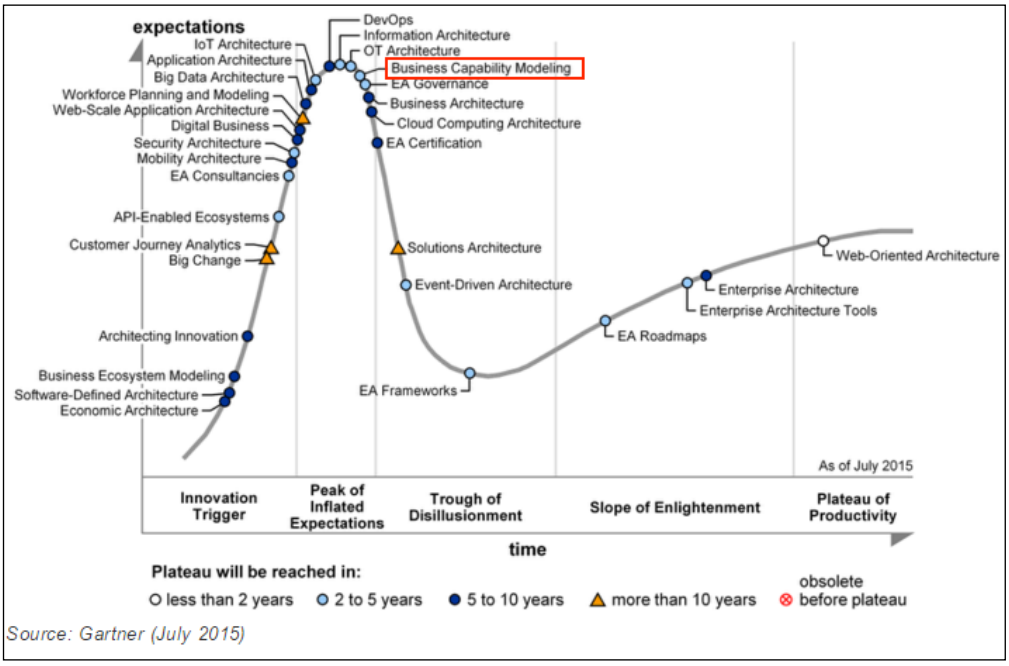


Figure 2.4.: Hype Cycle for Enterprise Architecture, 2015 [Gar15]

Business Capability Map is one of the most prominent publications on this topic. Their ten principles represent the scope of business capabilities [UR11]. First, they state that capabilities define *what* a business does, not *how* a business does something. Further, they are phrased in business terms, not in technical terms. They are also stable. That means capabilities are independent from the organizational model, technologies, and vendor solutions [FMSN11]. Furthermore, they are unique and thus not redundant. Capabilities have relationships to IT deployments and future-state IT architecture. Furthermore, automated capabilities are still business capabilities - not IT capabilities. Attention should also be paid to the fact that capabilities are of most value when incorporated into a larger view of an enterprise's ecosystem. To communicate the business capabilities a so-called business capability map has been established as a common approach. Put simply, it is a blueprint of the abilities for a given business. Business capability maps consists of multiple levels. As illustrated in figure 2.5, a capability map usually contains three hierarchically ordered levels. The highest level is called *foundation capability*, the second level are the *capability groups* and the lowest level of the map is referred to as *business capability* [UR11]. The Cutter Consortium presented two prin-

ciples for developing a meaningful business capability map. First, there must be only one map for a business which ensures a single point of truth. Furthermore, capabilities map to, but are not the same as, a line of business, business unit, business process or value stream. Although, the Cutter Consortium claims that capabilities have to be written as nouns [UR11] there is a disagreement in scientific literature. Cook (2007) for example argues that capabilities should be typed in noun-verb format (e.g., "generate invoice") [Coo07]. This approach seems to be more intuitive when talking about capabilities. Nevertheless, capabilities will be described as nouns in this thesis since the Cutter Consortium's Executive Report has often be quoted in other literature. In

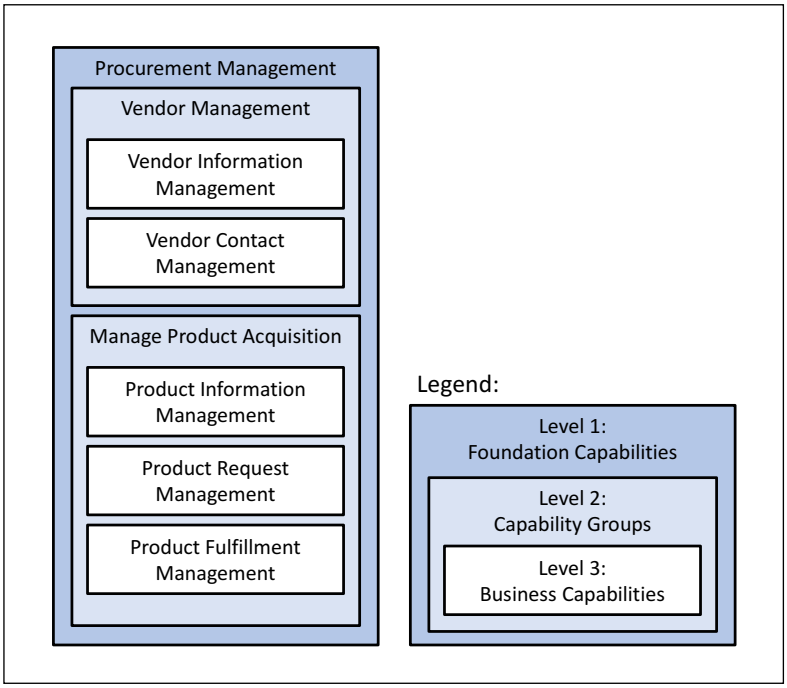


Figure 2.5.: Three-level capability decomposition [UR11]

traditional EA literature business capabilities are not included in EA model, in many cases not even mentioned at all [Kel09, EHHJ08, ARW08, Sch08, Nie06]. As indicated by Gartner's hype cycle, business capabilities have become an emerging concept in the last couple of years. Thus, there has been also conducted a lot of research regarding the relationship between EAM and business capabilities [FMSN11, BMP10, Pap14, Ber15]. The relevance of business capabilities and the benefits of capability maps are already proven by the dependency analysis of Andreas Freitag (2011) [FMSN11] and later in his dissertation where he analyzed the use of business capabilities during Mergers & Acquisitions (M&A) processes [Fre14]. The latest version TOGAF comprises a method for *capability-based planning* [Ver09]. Business capabilities in TOGAF are encapsulated into three dimensions: people, process, and material. The first dimension focuses on professional development and training. The process dimension includes concepts, business

processes and information management. The last dimension discusses the infrastructure components and equipment in general. In other literature, a similar structure can be found [HT06]. Beside the TOGAF extension there is also a framework presented by Microsoft which is called *Microsoft Services Business Architecture (MSBA)* [Mic08].
During the literature review on business capabilities there has been emerged parallels to the topics *Domain-Driven Design (DDD)* and *APM*. In general, DDD *"is a software development approach recommended to deal with complex and large-scale software systems"* [Mil15]. Unlike business capabilities, which are not directly part of the EAM field but rather a tool used to track the EA, DDD can be seen directly connected to EAM [WLR06]. In DDD software is developed based on a so-called *domain model* which is a abstraction of the business model [Eva04]. A domain model is always created by *domain experts* who usually has no IT background. Their responsibility includes only the understanding of business. After the domain experts agree on a domain model, it will be illustrated in a *context map* (see figure 2.6. The domain model contains multiple *bounded contexts*. They are defined as an independent part of the domain which are mostly assigned to one team [MA07, Sok15]. In practice, it is difficult to distinguish between a business capability and a bounded context. Some researcher even claim that it is the same thing [VdV15, VdL15]. Nevertheless, there is no clear differentiation between business capabilities and DDD. Further research on this issue would be desirable.
Obviously, the use of business capabilities is attended with a set of benefits. Eriksen listed three benefits of business capabilities [Eri10]:

- **Supporting technologies**
  This benefit is one of the reasons why Business Capability Modeling became so popular. Capabilities can be used to link to the underlying applications or technologies. In combination with the business capability map it can give an overview of the existing systems and their relationships to each other. Further, it reflects the business IT alignment proposed by Hevner (2004) [VAMPR04].

- **Technological risks**
  Based on the previous benefit technological risks such as applications with high failure rate can be identified. Furthermore, it helps to discover redundancies and thus streamline operations.

- **Investment alignment**
  Business capabilities also assist in aligning and allocating investments to the right capabilities. For that reason, the investment spend can be compared with the relevance of each business capability.

It should be noted, that there is striking similarity to the goals of APM (see section 2.3). Both try to display the status of the organization's capabilities (or applications) to assess their consistency with the business strategy. This perception will be a centric part of this thesis when it comes to the visualization of the application architecture via business capability map (see chapter 6).
The next chapter comprises publications and other scientific work which is related to this thesis.

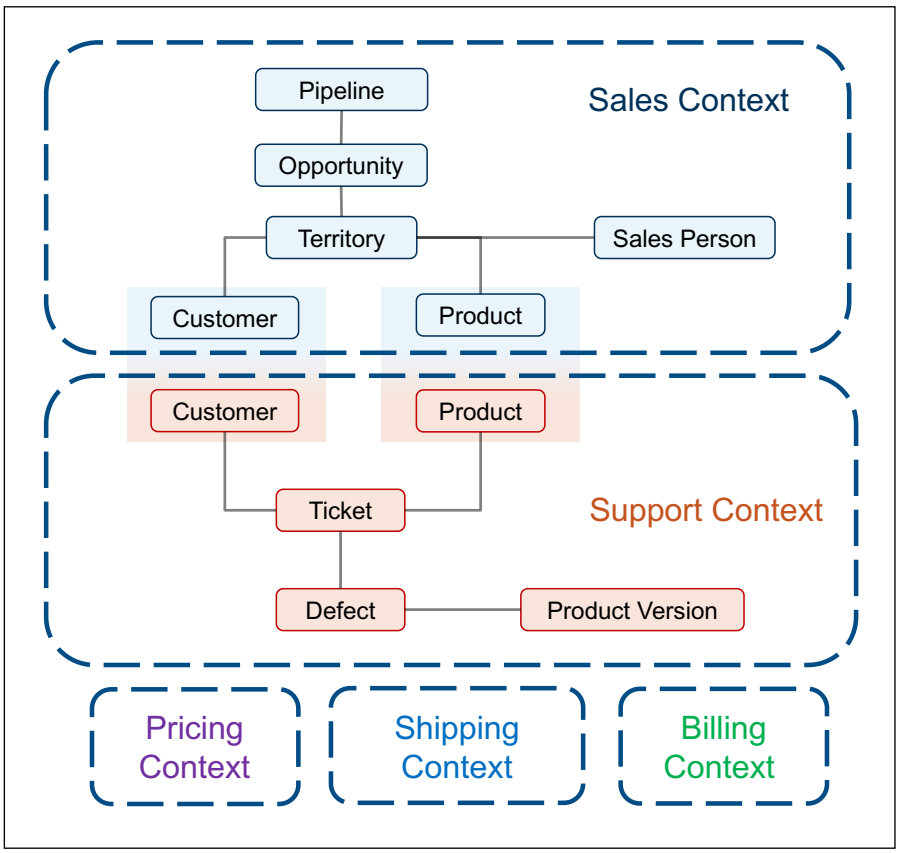


Figure 2.6.: Domain-Driven Design context map

3. Related Work

The following section reviews the prevalent literature on application architecture evaluation, application architecture visualization and their various determinants. After presenting the general approach of the literature review in section 3.1, the findings will be described in the next section. Further, the research gap and limitations of current literature regarding the underlying key objectives will be comprised in section 3.3.

3.1. Literature Review Approach

As mentioned in section 1.2, the research design methodology *literature review* was conducted in order to answer the research questions. This approach of viewing existing literature is also the first step towards identifying related literature. The scientific accuracy of the review is ensured by following the guidance for *Writing a Structured Literature Review* of Webster and Watson (2002) [WW02].
Although the branches APM and Business Capability Modeling are essential in IS and have much in common (see section 2.4), there is no literature found which investigates and compares both fields. On the other hand, there is a variety of literature providing insights for APM and Business Capability Modeling in an isolated way. Therefore, two different search queries are defined to identify relevant literature in each topic:

- **Search query 1:**
  "Application application evaluation" OR "Application application assessment" OR "Application application analysis" OR "Application portfolio analysis" OR "Application portfolio evaluation" OR "Application portfolio assessment" OR "Application landscape evaluation" OR "Application landscape assessment"

- **Search query 2:**
  "Business capability map" OR "Business capability" OR "Business capabilities"

Previously mentioned, in many cases terms are not precisely defined which leads to the use of the *OR* operator in the query. To ensure the sufficient quality and reliability of the reviewed literature, particular attention is paid to renowned journals. For that reason, especially journals of the *basket of eight* [1] and highly ranked journals by *Scimago Journal & Country Rank* [2] are taken into account. Even though this does not mean literature of other journals was excluded. In order to access those journals the scholarly online

[1] https://aisnet.org/?SeniorScholarBasket
[2] http://www.scimagojr.com

search catalogues are queried: EBSCOhost Online Research Database[3], ScienceDirect[4], Scopus[5], IEEE Xplore Digital Library[6], ACM Digital Library[7] and Google Scholar[8]. The following tables show the number of identified literature based on the search strings:

Search query 1: Number of information sources		
Database	**Search area**	**Count**
EBSCOhost Online Research Database	”TX All Text”	5
ScienceDirect	”All Fields”	6
Scopus	”All Fields”	25
IEEE Xplore Digital Library	”Full Text & Metadata”	0
ACM Digital Library	”Any Field”	9
Total		45

Table 3.1.: Search query 1: Overview of counted information sources

Search query 2: Number of information sources		
Database	**Search area**	**Count**
EBSCOhost Online Research Database	”Title only”	42
ScienceDirect	”Article Title, Abstract, Keywords”	25
Scopus	”Article Title, Abstract, Keywords”	241
IEEE Xplore Digital Library	”Metadata”	7
ACM Digital Library	”Any Field”	46
Total		361

Table 3.2.: Search query 2: Overview of counted information sources

In the next chapter the findings of literature review are presented.

[3]http://search.ebscohost.com
[4]http://www.sciencedirect.com
[5]https://www.scopus.com
[6]http://ieeexplore.ieee.org
[7]http://dl.acm.org
[8]https://scholar.google.de

3.2. Findings of Literature Review

With 45 literature identified, the results of the first search string are quite manageable. The second search query on the other hand, results in an enormous amount of literature including 361 potential sources. This can be seen as an indicator for the lately emerged relevance and importance of business capabilities. Admittedly, the entered string of the second query is compared to the first one very widespread. The reason for a widely varied search string is to ensure to completeness of the considered literature. The general intention of the thesis is to focus on application evaluation methods and extend it by a capability-based approach. Hence the first search query is distinct and the second one is wide.
During the literature analysis, it became quickly clear that the majority of the identified papers do not address the topic of the thesis. Some of them discuss the topic of business capabilities, some provide insights in application landscape or application portfolio evaluation but none of them provide a holistic view regarding the architecture's status of health. Neither they make use of the business capability map when assessing applications. Consequently, only a fraction of the identified literature is useful. Nevertheless, the review shows up some very interesting literature especially regarding the process of application portfolio rationalization [WV99, Sar06, FBvD07, Kro09, SFS10, GBB12]. Each of them provide a framework sharing the same structure: First, each single application is evaluated. While proposing this step, none of the mentioned literature provides a guidance *how* to evaluate the application. [WV99] argues to assess the holistic state of health, but neither he explains how to do it. The second step is to group the applications. Unfortunately, the grouping is always taken place on technical aspects. The categorization based on business capabilities had been left out. Finally, the category is assessed and a plan for action is provided. Again, further explanation of *how* to aggregate the application evaluating characteristics into KPIs measuring the status of the category is missing. There is also very few research on the topic of measure aggregating. Some attempts on KPI aggregation could be identified in the health and safety sector [J$^+$03, Pod15, SS08, JG07] but none in the IS sector. Although, KPIs form the central method in measuring and controlling IT systems there is a lack in research regarding the aggregation of performance indicators. The aggregation framework presented by Jollands (2003) [J$^+$03] will be used to develop aggregated KPIs in this thesis.
Some researcher also have conducted research on evaluating application portfolios. An interesting point is that a lot of attention is drawn towards application portfolio complexity [Moc09, SWK13, SWG13, LBMA14, SM14, SRSM15, Sch16, AKBA16, AKBMW16]. Mocker (2009) evaluated 273 applications of a bank on complexity. In order to measure complexity, he defined particular complexity driving application attributes. Subsequently, he found that interdependency-related complexity drivers are linearly correlated to operating costs. Another research on complexity correlation was conducted by Aleatrati Khosroshahi (2016) [AKBA16]. He analyzed the impact of listed application characteristics on operation costs and number of incidents based on application types (e.g., mainframe). The results show that operation costs and number

of incidents are positively correlating to the number of interfaces an application supports. This leads to the assumption that *number of interfaces* can affect the application architecture and thus will be considered for the KPI development. In another research Aleatrati Khosroshahi (2016) investigates the causes and consequences of application portfolio complexity. Eventually, 13 causes could be identified during a case study, which are resulting in five different consequences [AKBMW16]. A more holistic view on application portfolio complexity is provided by Schneider [SRSM15, Sch16] and Schütz [SWK13]. Both focus on heterogeneity of the application portfolio. By using topology- and entropy-based models they developed metrics which explain the cost of application portfolios. Resulting from a different thematic priority an evaluation beyond complexity is commonly missing. Nevertheless, this findings motivate the thesis to investigate application complexity in particular.
The second search query provided detailed insight into business capabilities in general. Especially the benefits of business capabilities in EAM are well studied [FMSN11, BMP10, FHU07]. Further it is shown that a business capability map is a very efficient tool to visualize and manage the set of business capabilities. It is also accepted that it can discover unnecessary redundancies and thus streamline operations and reduce operation costs. Additionally, it plays a part in improving an organization's business IT alignment [UR11]. Nevertheless, a guideline showing specific visualization methods for distinct use cases is missing. There is literature analyzing data visualization in particular [KLFK10, FSH14]. But overall, they do not fit the purpose of this thesis. Klinkmüller (2010), for example, illustrates the business capabilities solely from the business perspective. He uses a three-dimensional visualization which helps to *"identify capabilities of interest and understand the structure of the business"* [KLFK10]. The underlying IT components are not considered. Fittkau (2014) on the other side, visualizes large scale application landscapes purely on a technical level without relating them to business[9] [FSH14]. Both literature are scientifically accurate but thematically they focus either on business or IT. A combined view is missing. Therefore, the capability map can be very helpful where its current use lags far behind the possibilities which it possesses. In the next section the limitations of identified related work are summarized.

3.3. Limitations of Related Work

As indicated in the previous section, the literature synthesis revealed a set of lacks and limitations. Although there is intensive research conducted in APM a distinct guideline for evaluating an application architecture is missing. There is a variety of frameworks about *what* to do in order to assess the landscape but there is no guidance about *how*. A description in terms of what kind of application attributes in which form to choose is missing. Besides that, there is not sufficiently research done on developing KPIs for a holistic view. Although experimentally attempts exist in other scientific fields, further insights regarding IS would be worthwhile. In current research on application architec-

[9] https://www.explorviz.net

ture or application portfolio the main focus is on complexity. Further aspects are barely taken into account. However, insights on, for example, application architecture robustness would be helpful when it comes to evaluating the reliability of the landscape. Last but not least, the efficient and effective use of the business capability map is neglected so far. According to a study of the sebis chair, is the mapping of applicaitons to business capabilities one of the most relevant concerns in EAM [AKHSM15]. Therefore, the benefits of Business Capability Modeling cannot exploit its full potential.
Due to the fact that a holistic capability-based visualization of application architecture is missing, the need for this thesis is substantiated.

4. KPI Conceptualization for Application Architecture Evaluation

The following chapter is concerned with answering the first and third research question of the thesis. Thus, the main objective of this section is the development of KPIs in order to evaluate the application architecture and to propose recommendations to improve its status. Firstly, in section 4.1 the existing *Application Portfolio Rationalization (APR)* methods for application architecture evaluation will be presented. Subsequently, in section 4.2 application evaluating characteristics will be identified. Based on these insights, the KPIs will be created in section 4.3. The analysis of potential recommendations for action will be presented in section 4.4.

4.1. Application Portfolio Rationalization Methods

In order to achieve a company's objective every EA layer is equally important (see figure 2.1). A problem arising in the infrastructure & data layer affects all the layers above. Nevertheless, is the applications & databases layer a central part of EA. Most of the practitioners interact with applications in their daily work. That makes APM to a very practical and prominent research field with many scientifically perspectives. Consequently, a variety of research has been conducted regarding the process of APR [WV99, Sar06, FBvD07, Kro09, SFS10, GBB12]. Some researchers state that APR is used to *"assess the health of an information system's application portfolio"* [WV99] which comes very close to the objective of this theses. However, others call APR the *"redesign of the IS portfolio"* [GSK05]. This definition states the common view on APR. Therefore, it can be said that the purpose of APR and the purpose of this thesis slightly differ. Yet, there are similarities in the analysis of existing applications. Hence these methods form the foundation of the application evaluation process.
Figure 4.1 illustrates an overview about the existing APR methods. The relevant steps in order to answer the research questions are highlighted. Although, each process differs in detail, they can roughly be divided into the three phases: situation assessment, assessment evaluation and plan actions [FBvD07].

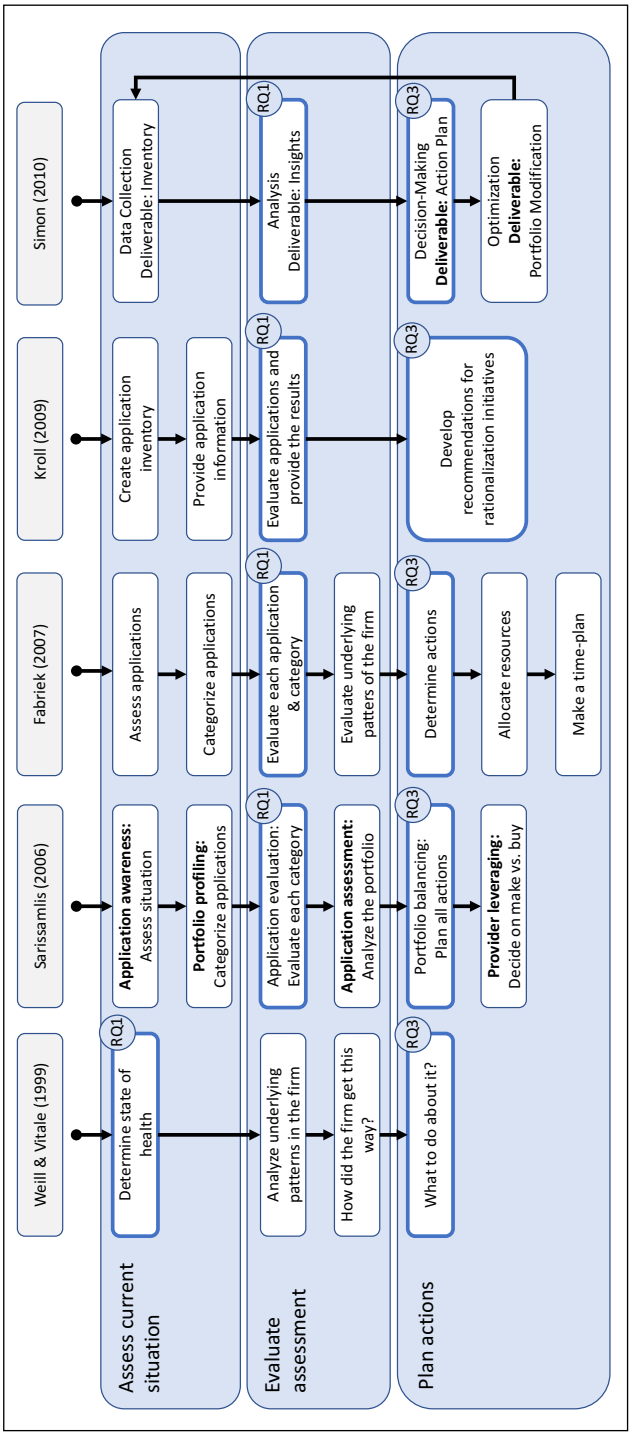


Figure 4.1.: APR methods comparison

- **Weill & Vitale (1999)** represent the oldest of the listed APR models. The other models which are developed several years later refer to the model of Weill & Vitale. Thus, it can be assumed that the recent models are refined versions of Weill & Vitale's APR model. The presented process consists of four steps. At the beginning the portfolio's state of health is defined. This very first step comprises the collection of application information and the evaluation itself. Weill & Vitale do not provide further insights of *how* to conduct the evaluation. Further, the patterns in the firm are analyzed and the reason of the portfolio status is questioned. Finally, health improving steps are proposed. This model comprises the essential steps of APR in a very high level. It lacks in detail and guidance how to achieve these steps.

- **Sarissamlis (2006)** focuses on reducing the complexity of the application portfolio. Compared to Weill & Vitale, Sarissamlis' model contains two more steps. Thus, the process is more detailed. Especially, the step *application categorizing* is relevant, since the mapping of application to business capabilities can also be seen as a kind of categorization. The remaining part is similar to Weill & Vitale's model.

- **Fabriek (2007)** recognized the similarities of the existing models and divided the APR models in three phases. In the assessment phase each application is not only reviewed separately, but also all together. In the next phase the assessment is evaluated. That means the metrics generated in the first phase are transformed into an interpreted application status (e.g., good status, bad status). Finally, the planning phase determines the actions that need to be taken in order to reduce application portfolio complexity.

- Compared to the previous APR methods, **Kroll (2009)** presents a simpler model with less steps. Nevertheless, the procedure is quite similar. The only difference is the missing categorization step. The applications are evaluated individually based on gathered application information. The last step is the development of recommendation for actions.

- **Simon (2010)** proposes the only iterative model. Besides he states that it is not only an APR model, but even can be seen as an APM model. Nevertheless, the steps are similar to the steps of the other models: Data collection, application analysis and creation of an action plan.

None of the presented APR methods seems to consider the involvement of business capabilities or the visualization of results. Consequently, inspired by the APR methods, a model for application architecture evaluation is created, which fits the purpose of this thesis. Figure 4.2 illustrates the steps which must proceed in order to evaluate the application architecture.
First, a list of relevant applications will be created. In the second step these applications are assigned to business capabilities. Based on the collected data, the application

architecture of the business capability will be evaluated. The status of the application architecture is illustrated in a business capability map in step five. Depending on the results, actions will be recommended in order to increase the application architecture status.
The selection of the relevant applications and their assigning to business capabilities is predefined by the industry partner and thus out of scope. In the third step the needed information for the evaluation must be gathered. But before doing this, the needed data has to be defined. For that reason, the next section comprises a literature review on application characteristics which measure the state of an application.

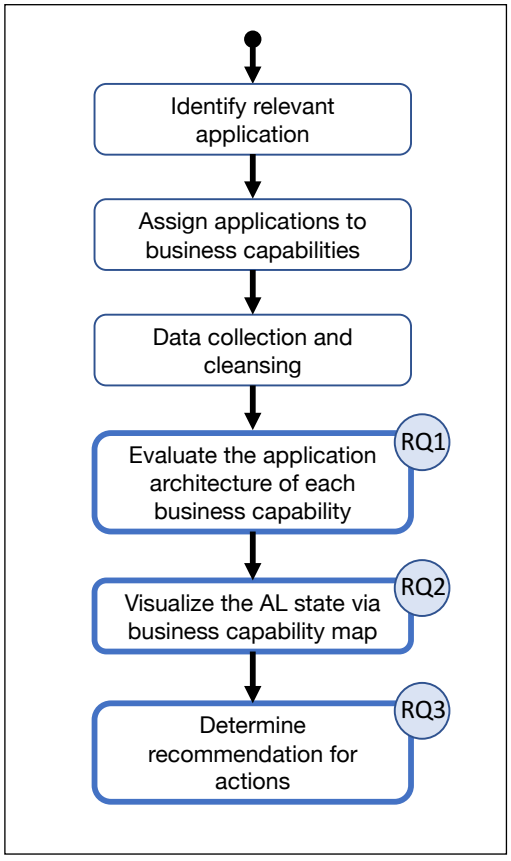


Figure 4.2.: Adapted method for application architecture evaluation

4.2. Application Characteristics

The following chapter primarily addresses research question one, which asks for appropriate application characteristics in order to evaluate an application architecture's status. However, this chapter also lays the foundation for the development of the KPIs. Nowadays, the landscape of larger enterprises embrace several hundreds of applications. New technologies and emerging trends in software development lead to increasing heterogeneity of the application portfolio. However, a method has to be applied in order to evaluate several applications on a common comparable basis. For this purpose a extensive literature review on appropriate application characteristics has been conducted. According to Bamberg (2009) in statistics a distinction is made between two different properties [BBK09]. *Qualitative properties* are observed and generally cannot be measured with a numerical result. *Quantity properties* on the other hand are numerical values in terms of a unit of measurement. Since KPIs are usually numeric values, the focus will be on quantitative application characteristics.
Based on the review of related literature and discussions with the industrial partner, a list of relevant application characteristics has been created. The results are illustrated in table 4.1.

Application Characteristics	
Characteristic	**Stated relevance**
Number of interfaces	[Moc09, Lan08, SRSM15, SWS+13, Ren15, KS03, AKBA16]
Capability coverage	[MST02, Moc09, SWS+13, Ren15]
Application age	[Moc09, Bee14, RWR06, Gro04, AKBA16]
Technological diversity	[Moc09, SWS+13, SRSM15, Ren15]
Functional coverage	[Lan08, SRSM15, SWS+13, Ren15]
Deviation from standard	[Moc09, BVVD09, SRSM15]
Application failure	[MST02, S.A14, VST07]
Application size	[BVVD09, MST02, VST07]
Functional overlap	[Moc09, SRSM15, Ren15]
Documentation	[BVVD09, MST02, S.A14]
Technology age	[BVVD09, Int12, Sch16]
Number of incidents	[Int12, S.A14, Lan08]
Operating costs	[Int12, Lan08], Industry partner
Number of users	[Lan08, AKBA16]
Business impact	Industry partner
Strategic relevance	Industry partner
Incident processing time	Industry partner

Table 4.1.: Overview of application attributes in IS literature

The **number of interfaces** of an application is listed as a measurable characteristic in six different sources. Thus, it is the most mentioned characteristic in literature. Gen-

erally described, it measures the interconnectedness of the applications in terms of interfaces they have with each other. According to literature this attribute is often used to measure the complexity of applications. The simple logic behind this hypothesis is the more interfaces one application has to other application, the higher is its complexity level. More interfaces mean decreased agility because the application cannot be adapted easily. In case of an application replacement this also means that the interfaces to other systems have to be tested. This attribute is described by the statistical data type *count* which is expressed in non-negative integers. Thus, it is classified as *cardinal* scale which is the typical level of measurement for quantitative values. In order to evaluate the interdependency, the number of each application's interfaces is counted up.
The **capability coverage** characteristic can be found in four different literature. It checks whether the application is used in multiple business capabilities. Many literatures refer to the usage in branches or departments, instead of capabilities. For the purpose of this thesis these findings are interpreted as business capabilities. Similar to *number of interfaces* the number of capabilities the application is used in is counted which means that the value is also classified as a *cardinal* scale.
Application age is one of the most reviewed characteristics in IS literature. Although labelled as unsuitable in some literature [SSM16] it is mentioned in five distinct sources as an important application attribute. It is an essential application information which is very likely to be available at many organizations. If not, it can be ascertained effortless. The application age is measured in years since its deployment. Like the attributes *number of interfaces* and *capability coverage* the measure of this characteristic is also cardinal scaled.
Technological diversity is mentioned in four distinct sources. It describes the number of different technology components the application runs on. One exemplary technology component is a database management system (e.g., MySQL). Like all the attributes mentioned so far, this is also a cardinal scaled measure which counts the existing components of the application. This characteristic is often found in literature reviewing the complexity of applications. Literature claims that a high number of components indicate complexity because of increasing heterogeneity of the landscape. Furthermore, different components imply higher maintenance effort which means higher operation costs.
The **functional coverage** is described as an application attribute in four literature. It covers the scope of functions which each application provides. It can be measured in numbers of business functions the application realizes [SWS+13, Sch16] or in a special unit of measurement called *function points* [Sch16]. In the function point method, the applications are assigned with a particular functional size. This unit is based on the functional requirements the application has to meet. Each requirement is assigned with a score depending on its relevance. Based on the number of fulfilled requirements the functional size of the application is added up. Since the scores for functional requirements are defined by the user, the results are very subjective. This makes it to a very abstract measure. The function size can be zero and each function size can be set in relation to another this measure. This makes it also cardinal scaled.

Deviation from standard deals with the compliance of the used technological components with the set of standard technologies within a company. IT standardization became an important part of EAM and thus is a major task of enterprise architects [Nie06, Kel12]. Company-wide IT standards are also endorsed by The Open Group [Ver09]. The most prominent expectations of IT standardization are cost reductions [DJ94] and interoperability [Tan05]. The downside of IT standardization are less optimal IT solutions due to predefined set of technologies [RWR06]. Since there is not a unified definition how to measure the deviation from a predefined standard, companies use different methods. Some can use qualitative methods like an arbitrary numerical scale where the application is mapped to a value based on its deviation. Exemplary, on a scale one to three an application can be mapped on one which means it is totally standard conform. This kind of measure is ordinal scaled because it allows only the ranked sorting. Further interpretations like the distance of each value to another is not possible. Thus, qualitative methods make it difficult to create a model which also contains values of quantitative methods. A better option to measure the deviation from standards is by using a percentage-based measure. Thereby the result would be cardinal scaled and thus more suitable for an equation.
With three sources of literature **application failure** shows an average reputation in APM. Put simply, it defines the reliability of the application by measuring its downtime [MST02]. That means the unit of measurement is *time*. It is, just like the majority of the attributes, cardinal scaled which results from the quantitative measurement approach. The most common unit which measures the downtime is *mean time between failures (MTBF)* which is originated from the field of supply chain management [Jon06]. The relevance of this attribute is beyond dispute. It illustrates clearly the robustness of an application and is thus one of the most important measures of the KPI. MTBF is cardinal scaled.
Application size represents a more technical application attribute. In total three sources of information could be synthesized which see the size of an application as an important attribute to evaluate software. All three are agreed that this attribute can be measured by the number of *lines of code (LOC)*. Logically, the size of the application is defined by its LOC. The more lines, the larger is the application. In his paper Vasconcelos (2005) states that the size of the application can also be measured by its *function points*. As mentioned earlier, function points actually measure the functional variety of an application. Following the logic that many functions consist of more LOC, Vasconcelos' hypothesis is comprehensible. However, three literature consider LOC as a suitable unit. Consequently, in this thesis LOC would be used to measure the application size. Both units are classified as a cardinal scale.
Function overlap deals with the functional redundancies of an application. It tests whether multiple applications support overlapping business processes. According to Ashkenas, many managers can name processes which they perform in different ways [Ash07]. Hence they may be supported by different systems. That can lead to applications covering certain functionality which is already covered by other applications [Moc09]. Exemplary, if multiple applications support the same business process for the

same product there is a functional overlap. This for sure results in unnecessary high IT expenses. The degree of functional overlap, thus redundancy of an application can be measured by simply counting the number of supporting functions of the application. The resulting unit is quantitative and belongs to the cardinal scale.
The attribute **documentation** basically assesses the availability of a documentation of the application. According to a study on application landscape conducted by the consultancy firm *Capgemini*, lack of documentation is a key challenge faced by IT architects. In order to measure the degree of documentation Morisio (2002) suggests to count the number of pages of documents associated to source code. On the one hand this indeed describes a quantitative approach with a cardinal scaled value but on the other hand the assumption that there is a correlation between the benefits and quality with the number of pages is debatable.
Technology age refers to the earlier mentioned attribute *technological diversity*. In total three sources state the relevance of modern technologies when evaluating applications. The most mentioned example for technology age in literature is respectively the age of programming languages. According to Bouwers (2009) who interviewed experts at the *Software Improvement Group* based in Netherlands [10], modern programming languages are normally preferred compared to older ones. One reason can be the syntax which is usually easier in modern programming languages. In order to measure this attribute simply the age of the programming language or platform can be calculated. As mentioned earlier the unit of measurement *age* is the result of a quantitative elicitation and thus cardinal.
In the *IT Infrastructure Library (ITIL)* an *incident* is defined as *"an unplanned interruption to an IT Service or reduction in the Quality of an IT Service"* [TLR07]. Since impacts may hinder the business it is safe to say that they are undesirable events which reflect the quality of the application. This very essential attribute can be measured by **number of incidents** which leads again to cardinal scaled values.
When talking about KPIs the first one which comes to mind may be **operating costs**. It reflects the origin of KPIs which is the domains of investment banking [Mar52]. But also in the field of APM the relevance of monetary KPIs is increasing [BAC11, Swa06]. This fact is also emphasized by the industry partner's particular interest in this attribute. Operating costs in general are expenses ensuring the operation of the business. Mapped to IT they are the costs in order to run the IT environment. This also includes business applications. Usually operating costs are measured in relation to a time frame (e.g., one year). Obviously, the unit it is measured in, is some kind of a monetary unit (e.g., euro) which is one of the most prominent examples for cardinal scaled numbers.
The **number of users** is an indicator for application importance. The more employees use the application, the bad is an application failure for the operative business. This attribute can usually be found in the EAM repository of many companies. Problems can occur when a small market with just a few employees is evaluated. In that case a distinct documentation of the number of users may be missing. Normally, the value found in the EAM repository describes the number of user related to the entire orga-

[10] https://www.sig.eu/

nization. One possible solution is to calculate the number of users relatively to the observed market. After calculating the percentage of potential users within the entire organization this insight can be mapped to the market. This approach results in cardinal scaled values. Another suggested unit of measurement for number of users are licenses the company owns [Lan08]. The challenge here is the heterogeneity in existing licensing forms which are billed differently. Thus, the comparability of the values is not possible.
In contrary to the mentioned applications attributes the attribute **incident processing time** is stated by the industrial partner only. It reflects the amount of time it takes in order to solve an incident. The relevance of this attribute was pointed out by the time aspect. Additional to the number of incidents it is important to know in which time an incident is solved. It makes a significant difference if an application generates five incidents which are solved in two hours or in two days. Similar to the age attribute or the underlying technology components this attribute is also cardinal scaled.
Besides operating costs, the **Business Impact Analysis (BIA)** is another monetary-driven attribute. It describes the impact of application failure. The question which it answers is: What are the consequences if application A does not run for the time period t. Thereby A denotes the specific application and t denotes the period of downtime [fSidI08]. Usually the result of a BIA is expressed in financial effects and thus in a monetary unit of measurement [Gar10]. This makes it to a quantitative approach with an cardinal scaled value. Just like the incident processing time this attribute is introduced by the industrial partner.
Strategic relevance is special positioned among the mentioned application attributes. It is also introduced during discussions with the industry partner. Strategic relevant applications are those applications which are particularly important to run the business. Its interpretation is similar to applications with a high business impact value. But this attribute allows a further strategy-oriented view based on the company's vision. Since many companies are interested in customer satisfaction in order to achieve long term success, those applications with a high strategic relevance may be applications which are connected and visible directly to the customer. Statistically this attribute is a binary data type. It defines whether the application is strategic relevant (value: 1) or not (value: 0). This kind of measure is very different to those mentioned in the thesis so far. The only possible interpretation which can be derived from that scale is the number of strategic relevant applications. A statement regarding "how important" the application is cannot be made. Measuring the strategic relevance for example, in monetary units may solve this problem. It that case a clear distinction between BIA and strategic relevance is needed.
The list presented in table 4.1 is an unfiltered illustration of application characteristics found in IS literature. This set already answers the first research question. However, in order to create KPIs this list is narrowed down during discussions with the industry partner. For this purpose, the opinions of several experts are taken into account. Interviews with people from different domains are conducted. On the one hand an enterprise architect has been asked for his opinion which attributes the most important are

and which he would rather disregard. The same question is applied to a business solution manager in the financial sector. The combination of professionals with IT related background such as the enterprise architect and business-oriented knowledge which is presented by the business solution manager leads to a very widespread view on the attributes. However, both experts are originated in the industry and thus are very practice focused. Therefore, also a research associate from the sebis chair of the Technische Universität München has been asked for his opinion on the application attributes. In this way, the academic accuracy of the thesis was ensured and an additional view from the scientific perspective was included into the development process of KPIs. Based on the feedback of those three experts a set of application attributes has been created which meets the needs of the domains IT and business. Further it allows a look at applications from the practice and scientific point of view.
After obtaining the opinions of the experts the following attributes have been removed:

- Functional coverage
- Application size
- Functional overlap
- Documentation
- Technology age

As mentioned the **functional coverage** is a quite fuzzy and abstract attribute. Additionally, the information gathering and creation of function sizes for each application involves major organizational outlay. The very same applies to *functional overlap* of applications. To identify functional redundant applications first they have to be mapped to business processes. As part of APM many companies may have already created such a map. But if not, this task takes a considerable amount of time and manpower which is associated with substantial costs. However, this attribute can be important for reducing the complexity and streamline the application architecture. Thus, it should be considered as part of the KPI in further research. **Application size** and **documentation** are both very technology-oriented application attributes. Since the KPI has to meet the requirements both, IT and business view it was decided to remove those from the list. Since the age of the application is already included in the list of attributes, it has been decided to take out the **technology age** of the list. The final list of application attributes which will be involved in the aggregated KPI is illustrated in table 4.2.

Application Characteristics	
Characteristic	**Stated relevance**
Number of interfaces	[Moc09, Lan08, SRSM15, SWS+13, Ren15, KS03, AKBA16]
Capability coverage	[MST02, Moc09, SWS+13, Ren15]
Application age	[Moc09, Bee14, RWR06, Gro04, AKBA16]
Technological diversity	[Moc09, SWS+13, SRSM15, Ren15]
Deviation from standard	[Moc09, BVVD09, SRSM15]
Application failure	[MST02, S.A14, VST07]
Number of incidents	[Int12, S.A14, Lan08]
Operating costs	[Int12, Lan08], Industry partner
Number of users	[Lan08, AKBA16]
Business impact	Industry partner
Strategic relevance	Industry partner
Incident processing time	Industry partner

Table 4.2.: Overview of KPI building application attributes

4.3. Application Landscape Evaluation

This section represents one of the key contributions of this thesis. In the following pages the process of KPI development as well as the KPIs by themselves will be presented. Firstly, the aggregation framework will be presented and the requirements for the KPIs discussed. Further it will be pointed out why the creation of one single KPI is not possible and what has been done instead. In the ensuing sections, each KPI will be described in detail.

4.3.1. KPI Requirements

First the meaning and definition of a KPI has to be clarified by conducting a literature review. David Parmenter (2011) did very promising research in the field of KPIs. His book *Developing, Implementing, and Using Winning KPIs* represents the most prominent scientific work on this topic. He describes KPIs as

> *"a set of measures focusing on those aspects of organizational performance that are the most critical for the current and future success of the organization."* [Par11]

In order to ensure the effectiveness of KPIs within an organization Parmenter also describes four *foundation stones for implementing KPIs*. The very first "stone" focuses on *partnership*. A successful performance measurement requires an effective relationship among stakeholders. From the senior management team to employees, to customers and suppliers. Each stakeholder has to recognize the acceptance and need for performance measurement. Further all have to be involved in a joint development of a strategy to introduce KPIs. The second important factor is *transferring power to the front line*. By empowering particularly employees which perform operational tasks in the "front

line" KPIs can be affected positively. By giving the staff decision-making permission they can act on critical circumstances more quickly. Parmenter's third foundation stone is to measure *only what matters*. Too many KPIs can distract the decision-making person from real problems. Thus, each KPI has to be linked to a critical success factor. An additional advantage of less KPIs is the maintenance effort which is connected to every single KPI. The collection and updates of the underlying data is usually associated with manpower. The last important stone is the *linkage of KPIs to the organization's strategy*. As mentioned before, this is achieved by directly linking the KPIs to critical success factors which are derived from the organization's strategy.
Furthermore, Parmenter defines seven characteristics of KPIs:

1. Are non-financial measures
2. Are measured frequently
3. Are acted on by the management team
4. Clearly indicate what action is required by staff
5. Are measures that tie responsibility down to a team
6. Have a significant impact
   (e.g., affects the critical success factors of the company)
7. They encourage appropriate action
   (e.g., they have been tested to ensure they have a positive impact on performance)

Since the listed characteristics have to be fulfilled by the developed KPIs, they can be considered as requirements. During discussions with the industrial partner this list was extended by three more requirements:

8. Are tractable
9. Are robust
10. Are comparable

For the industry partner it is important to understand the KPI and its realization. Consequently, the underlying equation has to be mathematical accurate but also traceable to the user. In the case of the industrial partner, the company has multiple locations of industry. Thus, the KPI has to be adaptable to different markets. This leads to the last requirement which is comparability. Those KPIs resulting from different markets have to be comparable with each other in order to assess each market relatively to each other. In the following section the development procedure of KPIs will be presented.

4.3.2. Approach for KPI Development

In order to develop *winning KPIs* Parmenter introduces a 12-step model, which is derived from the four foundation stones described in the previous section 4.3.1. According to Parmenter this model is based on findings from an extensive study conducted in cooperation with several organizations [Aus99].

1. **Senior management team commitment:**
   The very first step is obtaining the management's commitment to develop and use KPIs. This commitment is expressed by showing interest and taking actions like being available for the KPI developing team.

2. **Establishing a winning KPI project team:**
   Kaplan and Norton state that individually created KPIs with no or little management interaction have rarely been successful [KN96]. Thus, a well-trained team of experts in designing KPIs has to be established.

3. **Establishing a "just do it" culture and process:**
   During this phase the KPI developing team is encouraged to take action and create KPIs primarily by "just doing". Since every KPI has a unique objective, each KPI includes different attributes. Based on the fact, that the units of measurement of different attributes can strongly vary, a common guideline cannot be provided. Thus, the development of KPIs is a highly iterative process which is characterized by regular feedback discussions with experts. Due to the collected feedback, the KPI is improved step by step. That makes it to a very time-consuming procedure.

4. **Setting up a holistic KPI development strategy:**
   This step investigates the best way of implementation. This includes the definition of the KPI project team size, the creation of a timetable and further project management activities. The decisions made in this step primarily depend on the nature of the organization which is defined by its size, diversity of business units, locations and available resources. Consequently, the implementation is designed individually based on the company's properties.

5. **Marketing the KPI system to all employees:**
   Measuring performance within an organization is bounded up with personal obstacles. Employees responsible for the performance may feel themselves offended by the results of the KPI. Therefore, communication in beforehand is important. The employees have to be convinced of the need of KPIs and the resulting advantages.

6. **Identifying organization-wide critical success factors:**
   The development of KPIs is based on the organization's critical success factors. If not already done, in this step those factors have to be defined.

7. **Recording performance measures in a data base:**
   This step includes the identification and gathering of data based on which the KPIs are created. After collecting and cleaning they are stored in a database.

8. **Selecting team-level performance measures:**
   In order to ensure organization coherent behavior of each team, a performance measurement on team-level is created. This helps the team to clarify its objectives. Furthermore, performance measures on department-level, on division-level and on organization-level are developed. Parmenter does not provide further description on how to achieve that.

9. **Selecting organizational winning KPIs:**
   This step comprises the development of organization-wide KPIs. For that reason, the performance measures of teams, departments, divisions and the organization are analyzed iteratively. By consolidating the appropriate performance measures organization-wide KPIs are developed. The created KPIs have to ensure the in section 4.3.1 listed characteristics. This can be tested by applying the KPI on real data and checking its outcome for plausibility.

10. **Developing reporting framework for all levels:**
    This part deals with the illustration of the results. For that purpose, Parmenter recommends a dashboard view with up to 10 KPIs. The aim here is to establish a visualization which is meaningful and easy to understand. This topic will be discussed in greater detail in chapter 6.

11. **Facilitating the use of winning KPIs:**
    The developed KPIs can generate tangible benefits if they get widespread in the organization and become a part of its culture. This can be achieved only if the senior management team of an organization shows sufficient attention on the KPIs.

12. **Refining KPIs to maintain their relevance:**
    KPIs are not completed by developing them once. The KPI project team has to work on the KPIs continuously. By collecting feedback and suggestions for improvement the KPIs are refined. However, the stakeholders should be informed in case of a KPI modification. The reason for that are the changed values which otherwise can lead to misunderstandings.

It has to be mentioned that the steps for developing KPIs by Parmenter are not directly IS related. It represents a general approach on how to create KPIs in order to measure the performance in organizations. However, many steps of this methods can be adapted in the process of application architecture evaluation. Commitment of the senior managers is always key. The same is true for performance measurement in IT. Since the industrial partner is a big company with more than 100.000 employees, there is certainly no contact with senior managers during the thesis. However, the thesis is conducted in close cooperation with the group's EA division.
The second step is also important. Even though application architecture KPIs have a

more narrowed focus than organization-wide KPIs, a team of experts has to be established which develop the KPIs. Although Parmenter does not describe *how* exactly to create KPIs, it is obvious that KPIs do result of the right combination of data. This is achieved by equations. In order to ensure mathematical accuracy, the people within the team should show a sufficient understanding of mathematics and statistics. Summarized the ideal team member has to show knowledge in business, it and must be capable of handling with numbers. Obviously, the thesis and, by implication, the KPIs are conducted by the author of this thesis. Nevertheless, experts and the industrial partner are heavily involved in the iterative process of KPI development.
Performance measurement and KPIs have also attract attention in the field of IT. Thus, a lot of research was conducted on this topic [Küt11, KK07, Bro96, vdZ96, BR96]. In addition to scientific literature also the IT governance framework COBIT and the IT Infrastructure Library (ITIL) emphasize the importance of KPIs [Ins12, TLR07]. They also provide a set of recommended KPIs. But only a very few of them are related with applications or the application landscape. Most of them evaluate the IT systems as a whole. They provide KPIs for general IT processes, such as KPIs for IT investments or user satisfaction. KPIs evaluating the status of the application architecture are missing. Therefore, the "just do it" process is applied in order to develop such KPIs.
The fourth step, *setting up a holistic KPI development strategy*, is predefined by the examination regulations of the Technische Universität München. Beginning on June 15th, 2016 the duration of the developing process is six months. Based on the size of the organization a specific market is determined of which the application architecture is evaluated. Furthermore, a regular biweekly appointment with the industrial partner is arranged in order to get continuous feedback.
The results of this thesis have to be seen rather like a proof of concept then a productive system. Thus, the marketing of the KPI system to all employees was disregarded.
As mentioned KPIs usually are developed based on the company's critical success factors. For the industry partner the primary success factor is to have a throughout failure-free operations. From this organization-wide critical success factor an IT related success factor can be derived. It became clear that a failure-free operation can only be achieved by early detection of possible risks in the organization's application landscape in order to prevent issues. Thus, this states the success factor on which the developed KPIs are resulting from.
To perform the next step, first, relevant application attributes have to be identified. The procedure and results of this task are described in section 4.2. Since the thesis approaches only one market with a manageable amount of data, no database is required.
The next step comprises the creation of *team-level performance measures*. An evaluation on team-level is not in scope of this thesis. Still this step is very interesting. Parmenter states that the organization-wide KPIs which are developed in the next step, are resulting from the team-level performance measures. Thus, the basic approach is that organization-wide KPI are developed by consolidating appropriate performance measures of team-level and other low-level performance measures. By adapting and applying this approach to the field of capability-based APM the application architecture of

business capabilities can be evaluated. Put the case that team-level performance measures are replaced by individual applications and organization-wide KPIs stand for KPIs representing the application architecture as a whole. Consequently, as described by Parmenter, first the applications are evaluated and based on them an application architecture KPI will be created. Just as described in step nine, the development of the KPI is done by consolidating the performance measures of the relevant applications.
Part ten constitutes an essential part of the thesis. Even the second research question refers to this step. In order to identify a suitable reporting, possible visualization types are analyzed in chapter 6.
The last two steps deal with the ubiquity and maintenance of the KPIs in everyday business which is beyond the scope of this thesis.
Parmenter's approach on developing KPIs is clear structured and comprehensible. However, it lacks in precise description of *how* to create the *team-level performance measurement* and *how* consolidate those into *organization-wide KPIs*. Thus, further literature review is conducted to solve this problem. As a result, a variety of paper is found describing the procedure of performance measure aggregation. However, the identified literature was often worded in general terms or related to the health domain [J$^+$03, Pod15, SS08, JG07]. Here, too, in literature the question of *how* was not answered. However, a specific framework on calculating aggregated indices was presented by Jollands (2003) [J$^+$03]. In his publication he also identified the lack in a distinct framework for developing KPIs. Consequently, he presents a generic process which is illustrated by figure 4.3.

In the following each step is outlined in more detail:

1. Mapped this approach onto the evaluation of the application architecture, the so-called subindices are represented by application characteristics, identified in section 4.2.

2. In this step the appropriate application attributes are selected which are included in the target KPI. Jollands describes three considerations which have to be taken into account. First and most obviously, the selected attributes for aggregation have to share the same factors of interest. So, each attribute has to be a measure which indicates an interpretation of the same interest. Second, selected attributes must not be correlated. Otherwise this leads to the problem of *multicolinearity*. Due to multicolinearity the KPI would be biased towards the correlated attributes. Consequently, other attributes have less or no effect on the KPI at all. This problem is addressed by eliminating those variables that are correlated. Finally, a balance between statistical integrity and relevance to purpose has to be created. Often attributes there is policy interest in correlating attributes. Jollands mentions the example of energy generation and CO_2 emission. For decision-making purpose it makes sense to combine those measures although they obviously correlate. This results in the mentioned trade-off.

3. In order to create KPIs, the attributes somehow have to be aggregated. According to Ott (1978), in a mathematical perspective aggregation usually consists of sum-

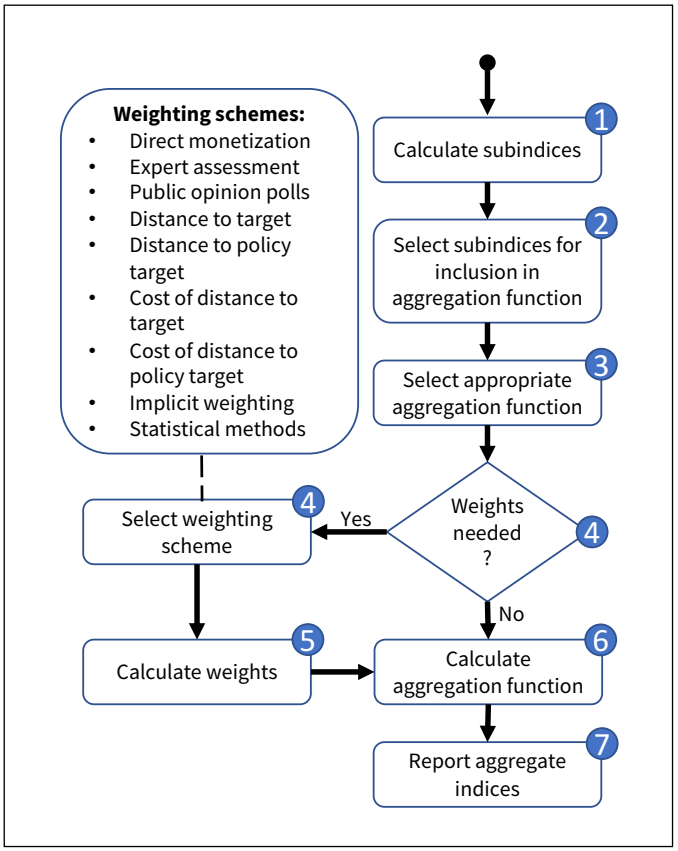


Figure 4.3.: Measurement aggregation process according to Jollands (2003) [J+03]

mation operation (attributes are added together), multiplication operation (product of attributes is formed) or maximum or minimum operation (reporting the maximum or minimum value of the attribute) [Ott78]. A combination of these operation within one function is possible.
Furthermore, in this phase four additional aspects have to be considered: First, the functional form of the attributes is crucial. Each attribute can either be of an increasing or decreasing-scale form. The increasing-scale indicates that higher values are regarded as a *worse* state than lower values. For decreasing-scale attributes it is the other way round.
When selecting the aggregation function one more aspect which has to consider is the equal impact of each attribute on the KPI. Ott (1978) has observed two potential problems with aggregation functions. The overestimation problem, where the KPI exceeds a critical level without any attribute exceeding it. In other words, the status KPI of the application architecture is ”bad” although the underlying application attributes are in a good shape. This problem can occur also the other way round. The underestimation problem implies that the KPI does not exceed

the critical level although some application attributes exceeding it. That means the KPI has a good value although the status of some individual application attributes is bad. Ott (1978) also mentions that this problem usually occurs with dichotomous subindices. Consequently, attributes can take on just two values (e.g., yes or no) should be avoided.
Often mathematically simple functions provide better results that complicated ones. The parsimony principle says that *"competing aggregation functions produce similar results with respect to overestimation and underestimation, the most appropriate function will be that which is the 'simplest' mathematically"* [J+03].
Furthermore, in scientific literature the requirement of divisibility came up. Hammond (1995) claims that after disaggregating the function into its components no information should be lost. That means the single parts of the function, which are the application attributes, have to be interpretable individually.
The last aspect which has to be considered is originated in the field of regression analysis. Similar to the development of KPIs, in regression analysis, a function is created. Based on historical data the model provides predictions on possible trends in the future. In regression analysis some of the mentioned aspects are also considered, such as multicolinearity. However, one important requirement when developing a regression function is the aspect of parsimony. It aims the development of a model, which is most efficient with the least number of terms [WJ03].
To provide an overview the mentioned aspects are consolidated in form of a checklist which is illustrated in table 4.3.

KPI Development Checklist	
Description	**Status**
The selected attributes share the same interest.	Yes / No
The selected attributes are not correlated to each other.	Yes / No
There are not too many attributes involved in the function.	Yes / No
Increasing and decreasing scale of the attributes has been taken into account.	Yes / No
The attributes have been tested for over- and underestimation.	Yes / No
The function is easily comprehensible and mathematically simple.	Yes / No
The function can be disaggregated to the separate components with no information loss.	Yes / No
The function meets the requirements (see section 4.3.1).	Yes / No

Table 4.3.: Checklist for developing KPIs

4. Weighting is an essential part of KPI development. Jollands has marked the weighting part as optional. However, it is highly recommended to use weightings in order to get more accurate KPIs. The selection of an appropriate weighting method is significantly important for the outcome. Inspired by [MBM+97], nine alternative weighting schemes are illustrated in figure 4.3. The cooperation with

an industrial partner allows a unique opportunity to interact with experts and access their opinions. For that reason, the weighting method *expert assessment* is chosen.

5. After selecting an appropriate weighting method, the weights are calculated and added to the function. This part will also be described in more detail in the subsequent sections 4.3.4 - 4.3.6.

6. Based on the selected aggregation function in step 3 and the weights in step 5, the final KPIs are designed. As mentioned by Parmenter, the calculation of the KPIs requires a data set of the identified application characteristics. This exemplary set is provided by the industrial partner. The anonymized results of this case study will be presented in chapter 5.

7. The results of the aggregation and thus the results of the KPI have to be reported. This step indicates the visualization of the KPIs, which was also included in Parmenters framework. The illustration options and a prototype will be shown in chapter 6.

4.3.3. KPI Categorization

The initial objective of this thesis is to create one single KPI describing the general status of health of the application architecture. In the early stages, it becomes clear that this approach is wedded to several problems. The consolidation of all application characteristics identified in section 4.2 would violate both aspects mentioned by Jollands, when selecting the appropriate subindices:

- **Same interest:**
  Each application characteristic describes a property of an application within a defined scope. The combination of all application characteristics would basically result in a loss of scope. The statement of the KPI would be ambiguous and cannot be interpreted. Thus, it is useless for decision-making.

- **Multicolinearity:**
  As mentioned, multicolinearity is the result of the combination of correlating application characteristics. During the literature review relevant paper was identified, which test such correlations. Both, Aleatrati Khosroshahi (2016) and Mocker (2009) conducted a research on correlation of application attributes regarding application landscape complexity. The results of the investigation are as follows: **Aleatrati Khosroshahi (2016):** The number of interfaces of an application correlates significantly with its operating costs and the number of reported incidents. Furthermore, there is a correlation between the number of incidents and the number of users, and between operating costs and number of users. When creating the KPIs these insights were taken into account. Thus, attempts have been made not to combine these application characteristics within one equation. However,

the characteristics number of users and operation costs are included in the same KPI. This attempt is justified by the weak correlation of the characteristics with a pearson rank of 0.19.
Mocker (2009): Mocker has identified a correlation between number of interfaces and the age of the application, number of users and operating costs. Furthermore, there is a positive correlation between number of DBMS and the application age and number of users. These results were also considered during the KPI development. But as Jollands described, there is always a trade-off to be made. Some attributes do match together from a business perspective although there is a correlation. As a result, the characteristics application age and number of interfaces are included in the same KPI.

Based on those insights and an iterative feedback process with industrial and scientific experts the application characteristics were divided into the following three categories. Each category represents one KPI:

Application Landscape KPIs		
Complexity	**Quality**	**Impact**
Number of interfaces	Application failure	Operating costs
Capability coverage	Number of incidents	Number of users
Application age	Incident processing time	Business impact
Technology diversity		Strategic relevance
Deviation from standard		

Table 4.4.: Application characteristics divided into categories

In the following sections three separate functions will be presented. Each function assesses the application architecture of a business capability in a different perspective which results in a KPI. However, as illustrated in figure 4.2, the application architecture of business capabilities is assessed by evaluating each application which is part of this capability. Further information about the proceeding will be described in the next sections. It has to be noticed that the aggregation takes place only for level 2 capabilities which is represented by *capability groups*. After discussions with the industrial partner it has been decided, that an aggregation on level 1 or drill-down on level 3 is out of scope.

4.3.4. KPI for Measuring the Application Architecture Complexity

The complexity of application architecture attracts much attention in the APM community [AKBA16, SRSM15, SWS+13, Moc09]. Therefore, it represents one of three KPIs developed during this thesis. Most of the application characteristics included in this KPI are derived from research paper focusing on application portfolio complexity. The selected application characteristics have all been identified by Mocker (2009) as complexity-driver. These findings are also confirmed by research of other scientists [AKBA16, SWS+13, SRSM15]. The relevance of each attribute can certainly be discussed. The characteristic *age* for example has been identified as unsuitable by [SM14]. However other researcher still include this attribute in their investigations [AKBA16]. Furthermore, there is always a trade-off to be made in terms of which data is available or easy to gather. Considering the principles of Jollands (2003) which are summarized in table 4.3, the following function is developed:

$$
\begin{aligned}
&Complexity_{c,m} = \\
&\frac{1}{|A_{c,m}|} \sum_{a \in A_{c,m}} \Big(\frac{number\_of\_covered\_capabilities_a}{\bigcup_{m \in M} total\_number\_of\_capabilities_m} \\
&+ \frac{(i\_interface\_in_a + i\_interface\_out_a) * P + e\_interface\_in_a + e\_interface\_out_a}{\bigcup_{m \in M} total\_interfaces\_in_m + \bigcup_{m \in M} total\_interfaces\_out_m} \\
&+ \frac{age_a}{\max(\bigcup_{m \in M} \{age_{a,m}\})} \\
&+ \frac{number\_of\_technology\_components_a}{\bigcup_{m \in M} total\_number\_of\_technology\_components_m} \\
&+ (1 - \frac{number\_of\_standard\_compliant\_technology\_components_a}{number\_of\_technology\_components_a}) \Big)
\end{aligned}
\tag{4.1}
$$

$$A_{c,m} := \{a | a \in A \wedge c \in C \wedge m \in M \wedge uses(c,a) \wedge located(c,m)\} \tag{4.2}$$

$$uses \subseteq C \times A \tag{4.3}$$

$$located \subseteq C \times M \tag{4.4}$$

Basically, there are three mathematical methods to aggregate the status of each application into a degree of complexity: arithmetic, geometric or harmonic mean [Cog86]. The *arithmetic mean* is mostly used for aggregating cardinal scaled values [BBK09]. This requirement is fulfilled since the majority of the selected application characteristics is a quantitative measure and thus cardinal scaled. The *geometric mean* is usually applied in time series analysis for successive expansions [BBK09]. Thus, this aggregation form is often used in the field of financial mathematics. One exemplary use case is the accumulation of interest rates. The *harmonic mean* approach addresses primarily the occurrence of outliers. By use of this approach the results would not be much biased, even in case of extreme outliers in the data set. For the scope of this thesis the arithmetic mean has been found the most suitable approach to aggregate application characteristics. The

reason is that the use case only contains data assigned to one distinct time period. Furthermore, the terms within the function are formed in a way, that no extreme outliers occur. Compared to the others the arithmetic mean is the most intuitive method which leads to greater traceability; one of the requirements of the industrial partner. Additionally, many relevant scientific works have also selected the approach of arithmetic mean for KPI aggregation [RGA07, KA04, SS08]. However, there are some aspects which are to consider when using the arithmetic mean. First, each term has to be non-negative. As mentioned, the values have to be within the same range. Outliers obviously would bias the result. Additionally, subindicators should have the same measurement unit [C$^+$08]. In order to meet these requirements some of the application attributes have to be transformed. These reasons and requirements count also for the Quality and Impact KPI which leads to the continuously use of the arithmetic mean and thus a consistent approach. In the following the function (4.1) for measuring the application architecture's degree of complexity will be described in detail.
This KPI represents the complexity level of a capability's c application architecture. The capability is located in a specific market m whereby the market m is a part of the organization's entirely locations M. Furthermore, a denotes an individual application from the set of all applications A. Consequently, $A_{c,m}$ denotes the set of applications used within the capability c of the market m. The relationships are noted according the mathematical set theory in equation 4.2 - 4.4. The union operator which occurs in the denominator describes the basis of the KPI.
The function consists of five parts where each represents one application characteristic. The first term describes the *capability coverage* of the application. The number of capabilities the application a is used in divided by the total number of capabilities of the market m results in a value between 0 and 1. The approach of dividing a value by the total amount of its kind has already been used in literature and established as reasonable [Ven08]. Subsequently, this method is also applied to create the Complexity KPI. In the worst case the application is used in every capability which means the ratio is 1. Additionally, the requirement of non-negativity is fulfilled since the denominator is a natural number. The union operator which occurs in the denominator ensures the previously mentioned KPI requirement *comparability*. In order to compare the status of different markets the KPIs has to be calculated based on the same data basis. This basis is achieved by merging the number of capabilities of both markets. This approach is illustrated by the union operator $\cup$.
The second part of the function covers the *number of interfaces* and thus the degree of interdependency of the application. First, the number of applications which the observed application a is connected with are added up. A distinction is made between incoming and outgoing data flow. As a consequent, both data steams are considered if an application receives and sends data to the same application. This leads to a higher application architecture complexity. During early discussions with the industry partner and scientists, it becomes clear that the number of interfaces is one of the most complexity-driving application characteristic. Furthermore, it was unanimously decided that there is a significant difference in complexity if the two connected applications are located

within the same capability or not. Consequently, a penalty variable P has been introduced in the function. For interfaces within the same capability a P value less than 1 is assigned. As it turned out during the case study a value of 0.5 is to be recommended. Like the previous term, the denominator denotes the number of all applications which are part of an interface. This set also includes applications which are not part of the market but have interfaces to market-applications. The highest level of complexity is achieved if the application a has an interface to each application listed in the denominator. In that case the term would be 1.
Age is a very special application characteristic. Compared to the others it is not a result of counted objects but years. However, years are also non-negative. In order to get a term which results in a value between 0 and 1 (just like the other terms), a denominator is introduced. The denominator is defined as the age of the oldest application within the market. That leads to a value of 1 for the oldest application in the market. However this approach comes to a plausibility issue. Imagine the oldest application in the portfolio is replaced. This should decrease the complexity of the portfolio. However, the oldest application is now a younger one and represents the denominator. Consequently, the overall value for complexity increases. Nevertheless, for evaluating the status quo of the application architecture the presented approach yields promising results. This arises from the result of the case study.
The role of the next application characteristic is easy to visualize. The more *technology components* an application uses in order to run, the more complex it is. Consequently, the higher is the overall complexity-degree of the architecture. To quantify this statement the number of the underlying technology components of the application a is added up. Divided by the variety of technology components within the market a ratio is created. An application which uses all the technology components listed in the market, would lead to a value of 1. This scenario reflects the worst case.
Last but not least, the deviation from company standards is included in the Complexity KPI. An application which does not meet the company standards leads to an increase in application architecture complexity [Moc09]. In order to quantify the degree of deviation insights from research conducted by Schneider (2016) are applied. He calculated the deviation by counting the number of standard compliant technology components of the application a. This divided by the total number of technology components, which are used to run the application, leads to a ratio which describes the compliance degree. Since we are interested in the opposite, such as, the deviation degree, this term is subtracted from 1. Consequently, the term results in 1 when all the components of the application are not confirmed by the company.
Summarized, the Complexity KPI consists of five subfunctions where each result in a certain value between 0 and 1. In the very worst case the result of the function would be 5.
In the next step, Jollands suggests to define weights for each subfunction to increase the accuracy of the result. Based on the idea that not every application characteristic represents complexity the same, weights are needed. The following table shows the results of the expert interviews regarding the characteristic's relevance.

Weights Complexity KPI	
Characteristic	**Weight**
Number of interfaces	0,47
Capability coverage	0,3
Application age	0,03
Technology diversity	0,1
Deviation from standard	0,1

Table 4.5.: Complexity weights based on expert interviews (n=3)

In total the opinion of three experts have been collected. Each expert was asked to scatter 10 points to each application attribute depending on its relevance in terms of driving complexity of an application. After all the scores of each characteristic are normalized so that they add up to 1. It has to be noted that there was not much deviation of each expert's opinion. Everyone has scored *number of interfaces* and *capability coverage* as very important. Both together account for 77% of the complexity degree. *Technology diversity* and *deviation from standard* each explain 10% of an application architecture's complexity. *Age* explains only 3%. This supports the result of research stating that age is not a suitable measure to define complexity [SM14].
Taking the weights into account (denoted as *g()*) the final Complexity KPI is formed as follows:

$$
\begin{aligned}
&Complexity_{c,m} = \\
&\frac{1}{|A_{c,m}|} \sum_{a \in A_{c,m}} (g(c) * \frac{number\_of\_covered\_capabilities_a}{\bigcup_{m \in M} total\_number\_of\_capabilities_m} \\
&+ g(i) * \frac{(i\_interface\_in_a + i\_interface\_out_a) * P + e\_interface\_in_a + e\_interface\_out_a}{\bigcup_{m \in M} total\_interfaces\_in_m + \bigcup_{m \in M} total\_interfaces\_out_m} \\
&+ g(a) * \frac{age_a}{\max(\bigcup_{m \in M} \{age_{a,m}\})} \\
&+ g(t) * \frac{number\_of\_technology\_components_a}{\bigcup_{m \in M} total\_number\_of\_technology\_components_m} \\
&+ g(d) * (1 - \frac{number\_of\_standard\_compliant\_technology\_components_a}{number\_of\_technology\_components_a}))
\end{aligned}
\tag{4.5}
$$

The last step, *reporting aggregate indices*, is described in chapter 6 which addresses the visualization of results in detail.

4.3.5. KPI for Measuring the Application Architecture Quality

The Quality KPI focuses on the application architecture's availability which indicates also its robustness. Based on the application characteristics *application failure*, *number of incidents* and *incident processing time* the following equation is defined:

$$\begin{aligned} &Quality_{c,m,t} = \\ &\frac{1}{|A_{c,m}|} \sum_{a \in A_{c,m}} (sum\_of\_downtimes_{a,t} \\ &+ (\sum_{k \in K} number\_of\_incidents_{a,k,t} * avg\_processing\_time_{a,k,t} * P)) \end{aligned} \tag{4.6}$$

This KPI quantifies the quality level of the application architecture of a capability *c* located in the market *m* during the period *t*.
The measures which quantify an application's availability very accurately are unexpected *downtimes* of the application. A downtime of 0 minutes indicates continuous availability and illustrates the best-case scenario. Vice versa a certain downtime limits the availability of the application. One important requirement of the arithmetic mean is the use of the same measurement unit within the equation. Based on the nature of the involved attributes the time unit has been selected as the most suitable. Thus, compared to the previous KPI no ratios are calculated. The subfunction of the KPI purely consists of the downtime in hours during the period *t*. In this case *t* is measured in years. However, this can be replaced by another unit like months.
In addition, the number of reported *incidents* are taken into account when developing this KPI. In ITIL an incident is defined as *"an unplanned interruption to an IT Service or reduction in the Quality of an IT Service"* [TLR07]. ITIL directly links incidents to the quality of an IT service. That indicates that incidents usually are mapped to IT services and not directly to applications. However, IT services can mostly be assigned to the involved applications which leads to the needed information on application level. During the biweekly jour fixes with experts from the industrial partner one important aspect has been crystallized when observing incidents. Usually incidents are categorized depending on their priority (denoted as *k*). Consequently, it is less bad when an application generates 10 facile incidents then 10 incidents of the category "high-priority". In the function this aspect has been taken into account by a penalty value *P*. Exemplary penalty values are listed in the following:

$$P = \begin{cases} 0,1 & \text{if } k \in incident_{low} \\ 0,2 & \text{if } k \in incident_{medium} \\ 0,5 & \text{if } k \in incident_{high} \\ 1 & \text{if } k \in incident_{critical} \end{cases} \tag{4.7}$$

During the case study a maximum *P* value of 1 for a *critical* incident has been established as reasonable. This value indicates that an incident of the category *critical*

compromises the quality of an application as much as an unexpected downtime.
Furthermore, the processing time of an incident is also identified as a relevant characteristic. The longer it takes to solve an incident, the longer the application is affected by the incident and the less is its quality. Especially, the combination of a high-priority incident with a long processing time can be a big problem. In order to transform the *number of incidents* into a time unit, it was multiplied by the average processing time of the incidents from the category *k*.
Subsequently, the weights are presented which are identified during expert interviews.

Weights Quality KPI	
Characteristic	**Weight**
Application failure	0,53
Number of incidents	0,33
Incident processing time	0,13

Table 4.6.: Quality weights based on expert interviews (n=3)

The experts were on the same page regarding the relevance of the application attributes. The characteristic *application failure* has been identified as the most important one. According to the interviews it explains more than the half of an applications quality. The importance of incidents add up to 46%. Composing 33% of the *number of incidents* and 13% of the *processing time*.
Including the weights, this is the final function which describes the Quality KPI:

$$
\begin{aligned}
&Quality_{c,m,t} = \\
&\frac{1}{|A_{c,m}|} \sum_{a \in A_{c,m}} (g(d) * sum\_of\_downtimes_{a,t} \\
&+ (g(in) + g(t)) * (\sum_{k \in K} number\_of\_incidents_{a,k,t} * avg\_processing\_time_{a,k,t} * P))
\end{aligned}
\tag{4.8}
$$

The weights of *number of incidents* and *average processing time* are added up in order to ensure the mathematical accuracy. However, since the weights cannot be distinguished, this leads to a decrease in accuracy of the result.

4.3.6. KPI for Measuring the Impact of Application Architecture Failure

The software-induced plane crash in 1993 shows that the failure of applications can have a massive impact [Brä93]. If not necessarily leading to casualties, application failure can also be fatal for the business. According to an industry survey conducted by Gartner Inc., a network failure can cause costs of $300,000 per hour [Gar14]. Consequently, failure of critical applications can cause a significant financial impact. Selecting the appropriate application characteristics, the function for the Impact KPI is as follows:

$$
\begin{aligned}
&Impact_{c,m,t} = \\
&\frac{1}{|A_{c,m}|} \sum_{a \in A_{c,m}} \Big(\frac{operating\_costs_{a,t}}{\max(\bigcup_{m \in M} \{operating\_costs_{a,m,t}\})} \\
&+ \frac{number\_of\_users_{a,t}}{number\_of\_employees_{m,t}} \\
&+ critical\_business\_impact_a \\
&+ strategic\_relevance_a \Big)
\end{aligned}
\tag{4.9}
$$

This function calculates a value for impact of application failure for the capability *c* in the market *m* during a given period of time *t*.
The *operating costs* of the application are a central part of the equation. For the business related expert, who is working for the industrial partner, this attribute represents one of the most important measures to include in this KPI. Operating costs are paid anyway regardless the application runs properly or not. As a consequent, the more expensive an application is, the higher are the costs which are caused by the failure. Obviously, the costs are measured in a monetary unit. However, they have to be transformed, since Parmenter defines in his KPI requirements that KPIs should be *non-financial* measures. By dividing the operational cost based on the maximum, which also came into effect in the Complexity KPIs, a value between 0 and 1 is resulting. In order to compare different markets the $\cup$-operator is used.
The next characteristic describes the *number of affected users*. If the application is used my many users, its failure is worse than when it is used by just a few employees. Consequently, if employees cannot work the business is affected directly. By dividing the number of people using the application by the total number of employees in the market, a value between 0 and 1 is calculated. 1 means that everybody in the observed marked is affected by the failure and 0 means nobody is affected.
The *business impact* characteristic describes the caused costs of an application failure during a certain time period. Usually this is calculated within the scope of a BIA where the result is displayed as a monetary unit. During the case study the problem of missing information occurred. Due to this and the limited time it was decided to apply another approach in order to measure this characteristic. A list of application is presented which are identified as critical regarding the business impact. Hence this value is ”yes” or ”no”. Either the failure of the application has a high business impact or not. Compared to the other application characteristics, which are all presented in a cardinal

scale, this characteristic is obviously binary data. This leads to biased results using the arithmetic mean in order to map the application status to capabilities. However, the arithmetic mean method is also used to calculate the Impact KPI. This approach is explained by a trade-off to keep one consistent approach for each KPI equations. In future research this term can be replaced by one including the business impact in a monetary measurement unit. This can look similar to the term which describes the operating costs.
Furthermore, the *strategic relevance* of an application is identified as an important characteristic during discussions with experts. A as strategic relevant labeled application should run continuously. A failure of such an application can lead to consequences in the business. This does not necessarily mean that it would cause high costs, which is within the focus of the *business impact* attribute. A strategic relevant application, for instance can be some kind of a decision-supporting application used by the senior management. Compared to an application responsible for robotics on the assembly line, the failure of this application would not immediately cause costs. Just like the *business impact* characteristic this attribute is also described in binary data. A range between, for example, 1 and 10, with 1 for low strategic relevance and 10 denotes the highest degree for strategic relevance would lead to less biased results.
Summarized, the logic behind the Impact KPI is the same as the Complexity KPI. Each subfunction can take values between 0 and 1. In the wost case each one would result in a 1 and in the best case the values would be 0. In the following the weights (table 4.7) based on expert opinions will be discussed.

Weights Impact KPI	
Characteristic	**Weight**
Operating costs	0,33
Number of users	0,1
Business impact	0,3
Strategic relevance	0,27

Table 4.7.: Impact weights based on expert interviews (n=3)

The interviewed experts agreed on the higher relevance of *operating costs*. This attribute explains 33% of the failure impact. Followed by *business impact* which explains 30%. With a slight difference of 3% the application's *strategic relevance* is almost on the same weight. *Number of users* placed a distant fourth in the scorings. It is only responsible for 10% of an impact. Besides higher accuracy the weightings can also weaken the bias caused by the binary data of business impact and strategic relevance. The less the weights for those two characteristics are, the low is the affect of an extreme value like 1 on the final result. Taking the weighing into account the final function describing the impact of an application architecture's failure is illustrated in the following equation.

$$
\begin{aligned}
&Impact_{c,m,t} = \\
&\frac{1}{|A_{c,m}|} \sum_{a \in A_{c,m}} (g(c) * (\frac{operating\_costs_{a,t}}{\max(\bigcup_{m \in M} \{operating\_costs_{a,m,t}\})}) \\
&+ g(u) * \frac{number\_of\_users_{a,t}}{number\_of\_employees_{m,t}} \\
&+ g(b) * critical\_business\_impact_{a} \\
&+ g(s) * strategic\_relevance_{a})
\end{aligned}
\tag{4.10}
$$

4.4. Recommendations for Action

This chapter primarily discusses the third research question. It proposes potential actions in order to improve the application architecture status. To start off with, the recommendations are made on the assumption that each application characteristic does not influence each other. In fact, this is a more theoretical approach since in the field there are restrictions. For example, the operating costs and number of users are correlated [AKBA16]. Thus, the operating costs cannot be reduced and while the same number of employees use the application. Metaphorically speaking, you cannot cut software licences and yet provide the application to the same number of users. However in the following sections theoretic recommendations for actions will be discussed.

4.4.1. Reduction of Application Architecture Complexity

The Complexity KPI basically quantifies the degree of complexity of the application architecture. In order to reduce complexity, first, the nature of the KPI has to be analysed. According to Jollands, one of the aspects which has to be considered is the scaling of the KPI. Based on the attributes it can be said that this KPI is increasing scaled. That means the higher its value, the higher is the complexity. Consequently, in order to reduce the complexity the value of the KPI has to be decreased. To do that the value of each subfunction and thus the values of the attributes have to be reduced.

- **Capability coverage**
  The value of the first subfunction is based on the number of business capabilities the application is used in. Consequently, if an application is dedicated to one single business capability it would result in a minimum possible value. If there is an application which is not related to any business capability it can be concluded that this particular application is redundant and probably not business relevant. After discussing with the right application owner, it may be eliminated.

- **Number of interfaces**
  The more interfaces an application has, the higher the application architecture complexity [Moc09]. Thus, the number of connections to other applications have

to be reduced. Resulting from the Complexity KPI there is a distinction between business capability internal connections. These kind of interfaces are preferable since the involved applications are within the same capability. At this point there is another trade-off to be made. It is traceable that few interfaces lead to less complexity. However, companies try to split their core systems in order to increase flexibility. Resulting in multiple individually manageable modules the overall agility may be increased. But this approach leads to more interfaces. For instance, a core system initially has n interfaces. Assuming this system is divided in m modules, the resulting number of interfaces would be $n * m$. Consequently, the number of interfaces has increased by a factor of m.

- **Application age**
  In order to decrease the value of this subfunction, the existing application has to be replaced by a new one. Here an essential condition is that the new application has to fulfill the functions covered by the previous application.

- **Technology diversity**
  Just like the previous subfuntions, the value of this one has to be decreased. For that reason, the number of underlying technological components has to be reduced. This can be achieved, for instance, by replacing the application.

- **Deviation from standard**
  Last but not least, the deviation from standard can be reduced by deploying company consistent technologies. Consequently, when acquiring a new application, the list of confirmed technologies has to be taken into account.

4.4.2. Increase in Application Architecture Quality

Just like the Complexity KPI, the Quality KPI and its subfunctions are increasing scaled. That means the overall quality of the application architecture would increase if the attribute values decrease.

- **Application failure**
  It is traceable that the sum of application downtimes is representing the quality of an application. The more failures, the less the application is available and obstruct the business. Sometimes application failure is not preventable. In that case, the support team which is responsible for the recovery has to act quickly. This point complies with the second *foundation stone for implementing KPIs* presented by Parmenter (2011). He states that employees who perform operational tasks should act autonomously. This reduces the time to fix the problem.

- **Number of incidents**
  The more incidents are generated by an application, the less is its quality. In order to increase the applications quality, the number of incidents has to be reduced. Additionally, the distinction between incident types has been made. This logically

leads to the fact that incidents of the category low are more preferable than critical incidents. For that reason, elimination of critical incidents has high-priority.

- **Incident processing time**
  The recommendation in order to reduce the incident processing time is the same as presented for application failure. The faster an incident is solved, the less it affects the course of business.

4.4.3. Avoid serious consequences of Application Landscape Failure

Sometimes application failure cannot be prevented. Thus, the monitoring of failure consequences is very important. Like the other KPIs, the Impact KPI and the involved application characteristics are increase scaled. Thus, the values of subfuncitons have to reduced in order to avoid serious consequences.

- **Operating costs**
  The most intuitive way to reduce operating costs is by eliminating redundant applications. In many cases this is not easily viable. However, in some cases it can make sense to replace traditional applications by cloud-based solutions according to the *software as a service* model. Especially, if no sensitive data is involved this approach can be an alternative. However, mostly applications are locally hosted and thus imply IT infrastructure costs. Gartner identified several actions to reduce these costs. By server virtualization for instance energy costs can be reduced by 50% [Gar11]. Modernizing or consolidating data centers are further options to reduce overall operating costs. Pushing down IT support is one more suggestion to reduce costs. This recommendation however is in conflict with the suggestion of increase IT support in order to reduce incident processing time. Based on the priorities a trade-off has to be made.

- **Number of users**
  A high number of users affected by an application failure can lead to a lack in efficiency. However, this subfunction is barely controllable. In order to decrease this value the number of active users of an application has to be reduced. It is difficult to hinder people in using a fundamental application. One possibility is rolling out an additional application which fulfills the same functions. However, this approach would increase complexity and imply additional operating effort.

- **Business impact**
  Like controlling the number of users, it is difficult to reduce an applications business impact. One way may be additional redundancy. Since it causes additional costs, *redundancy* is a word with negative connotations. Through redundant applications and IT components the robustness of an application would increase and thus the probability of a failure would be reduced. It may make sense to apply this strategy on applications with a high business impact.

- **Strategic relevance**
  Whether an application is strategic relevant or not is usually predefined. Consequently, in order to reduce the value of the Impact KPI it is preferable to reduce the number of strategic relevant application to a minimum.

5. Case Study

In this chapter the developed equations are applied to real data provided by the industrial partner. The main purpose is to create the groundwork for the evaluations in chapter 7. Based on the results of the evaluation the accuracy of the KPI outcomes will be determined.

5.1. Data Gathering

The industrial partner is a globally operating automobile company with its head office in Europe. With an application portfolio of approximately 5,000 different applications it requires a structured overview about their application architecture's status. Hereby an application is defined by the automobile company as a distinct executable software program. This does not involve Microsoft Excel based macros. This heterogeneous application portfolio which includes an extraordinary number of applications is considered suitable for a case study. Due to the time limit of the thesis and primary purpose of evaluating the accuracy of the equations, it was decided to focus on a manageable selection of applications. In total 6 potential markets have been analyzed based on data availability, accessibility and the size of the data set. As a result, a suitable market is identified.
With less than 100 employees the observed market is one of the smallest locations of the company. It is located in South Europe and is primarily responsible for financial service of the automobile company. The business is based on 12 level-1 business capabilities which rank among 7 level-0 business capabilities. In total the application portfolio comprises approximately 30 applications.

5.1.1. Data Collection and Cleansing

Although, a majority of the application data is easily assessable from the company's EAM repository, it was an activity of several months to collect and clean the necessary data. In order to keep track of the information regarding each application Microsoft Excel is used. All the information regarding the business capabilities and applications is consolidated in one file. In the following the collecting and cleansing procedure for each KPI is described:

- **Complexity KPI**
  Since the function of the Complexity KPI is the most comprehensive one, the data preparation is a long process. In order to calculate the complexity status data on the associated application characteristics are gathered. The required data for

capability coverage and *number of technological components* is obtained from the industrial partner's EAM repository. The *age* of the applications is calculated in years based on the release date. The identification of capability internal and external interfaces however requires a substantial manual effort. For this purpose, each business capability and its connected applications are analyzed individually. The *penalty value* for internal interfaces is defined as 0,5.

- **Quality KPI**
  The major application characteristic of the Quality KPI is the number of reported incidents. As mentioned in the earlier sections, incidents usually are not mapped directly to applications. This applies also to the industrial partner. However, for the calculation of the Quality KPI the number of incidents for each application is needed. To solve this problem the incidents are mapped over IT services to the applications. With the help of a list containing the applications involved in each service this is possible. In many cases one service contains multiple applications. In this instance the number of incidents is divided equally into the applications. If the application is represented in multiple services, the number of incidents of each service is added up. The *incident processing time* is calculated based on the difference between the date it is submitted and resolved. Afterwards the average processing time for each impact category was identified. The *penalty value* for each category is defined based on the opinion of an expert from the industrial partner. The exemplary selected values are the same as displayed in the equation 4.7. It has to be mentioned that the list of reported incidents and the information about the processing time was not in the EAM repository and thus gathered from an external source.

- **Impact KPI**
  The *operating costs* which is an essential part of the Impact KPI was collected in multiple iterations. It turned out that the information collected in the first place contained only the costs of the applications directly related to the observed market. After a meeting with experts responsible for operating costs the remaining data could be identified. The costs of applications used by multiple markets are basically calculated by dividing the total costs into the number of sharing markets. This approach is selected with regard to the limited time. A more accurate outcome can be obtained by weightings based on the intensity each market uses the application. The *number of users* is originally given in 4 different ranges. In order to map this ranges into distinct vales first a distribution on percentage basis is to derived. The fourth range which represents the largest group of users is set as 100% affected users. The other ranges are rated based on their limits. Subsequently, this percentages are multiplied with 42 which is the number of employees located in the observed market. The list of applications with a *critical business impact* is created during the processing time of this thesis.

Despite every effort the data on *deviation from standard*, *downtimes* and *strategic relevance* could not be gathered until the submission of this thesis. These subfunctions are

temporarily left blank. The gathered data is related to the calendar year 2016. Consequently, the data of some characteristics are not complete (e.g., number of incidents) since the data set was created on November 15, 2016. The operating costs are based on forecasts for 2016. However the majority of the data is relatively static and does not change frequently (e.g., number of covered capabilities and technology components).

5.1.2. Multicollinearity Test

As suggested by Jollands (2003) the selected subfuncitons within an equation should not be correlated. Otherwise the results can be biased. Mocker (2009) and Aleatrati Khosroshahi (2016) have already conducted research on the correlation of application characteristics. The developed KPIs contain attributes which are not considered by these papers. Consequently, each characteristic within the same KPI is tested on correlation in order to avoid multicollinearity. The test was applied on the data set provided by the industrial partner. The test was conducted by using the statistical software environment *R*. The complete outcome is displayed in the appendix.
The generally very high p-values indicate no significant correlation between application characteristics. However, the p-value of 0,01 may mean that the *number of interfaces* correlates with *number of technology components*. Furthermore, a significant correlation between the *number of high incidents* and *critical incidents* was identified.
Admittedly the sample provided by the industry partner, which includes information for approximately 30 applications is too small for reliable results. Nevertheless, this results inspire to conduct another test based on a lager data set in the future.

5.2. Application Architecture Evaluation

In the following the application architecture of the observed market is evaluated. The industrial partner has access to the results of the complete evaluation of the observed market by running the prototype. Below, the hot spots of the market will be described in detail. The business capabilities and applications are encoded according to the non-disclosure agreement. In summary, it can be said that the application architecture of 6 business capabilities is in a critical state. That is 50% of the entire set of business capabilities. However, the critical application architectures are divided in different categories. The distribution is illustrated using a Venn diagram (figure 5.1). Each business capability and application is anonymized by use of codes such as BC1 or APP1.
Referring to the degree of **complexity**, three business capability's application architecture has been identified as critical. This complexity results from the included applications. Interesting is that the complexity of the business capabilities BC1, BC2 and BC2 is caused by one single application. The application APP1 is involved in each of the business capabilities. In consequence, the business capabilities are displayed in a *red* status. Drilling further down shows why this application is marked as complex. One of the reasons is, that it is used in 8 of 12 capabilities. Furthermore, it has 11 interfaces to other applications and it is based on seven technology components. Only the age of

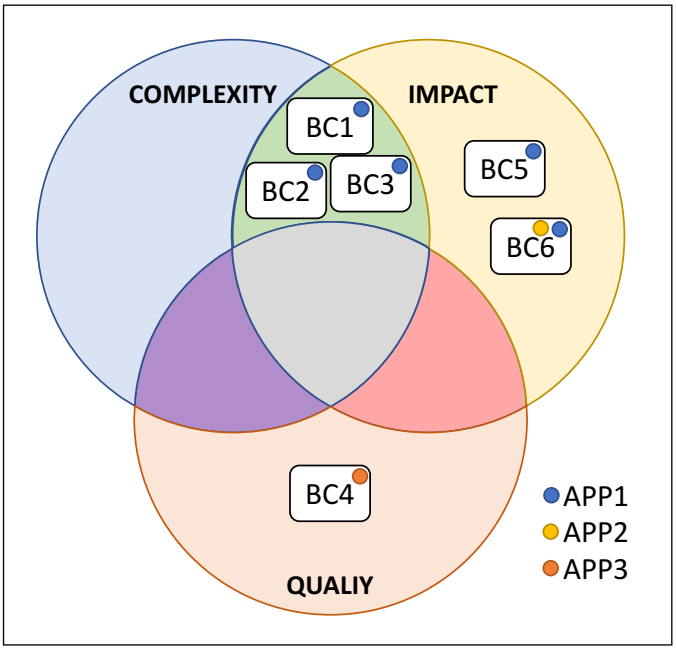


Figure 5.1.: Business capabilities with critical application architecture

two years is on an acceptable level.
The overall **quality** of the application architectures is in a good condition. Only BC4 shows a deficit in quality. This naturally results from the involved applications. Here application APP3 is in an extraordinary bad shape. It caused 73 incidents of the category high and 65 of the category critical in 2016. Furthermore, the average processing time for incidents of the mentioned categories is approximately 40 hours. Although weighted low by the penalty value the average time to resolve a incident of the category low is 369 hours.
According to the outcome of the **Impact KPI**, five business capabilities are at risk in case of application failure. Since these capabilities represent 42% of the whole set of business capabilities, this is a quite alarming situation for the market. The main reason for this outcome is again the application APP1. It not only drives the degree of complexity, but also is responsible for the critical state of business capabilities with regard to failure impact. This application is also the reason why the critical business capabilities BC1, BC2 and BC3 overlap in two categories as illustrated in figure 5.1. Furthermore, APP1 is also contributing for the poor state of BC5 and BC6. The main reason why this application is marked critical is its high business impact. During a BIA it was listed as an application with an high impact. Additionally, it causes relatively high operating costs, what makes sense, since it is involved in many business capabilities. The other critical application is APP2. It is part of BC6 and thus increases its impact value. Compared to APP1 this application has an even worse state. Additionally, to high business impact and high operating costs its failure would affect many employees.

5.3. Recommendations for Action

In the previous section the hot spots of the observed market are described. Now the generally discussed methods to improve an application architecture's status in section 4.4 will be applied to this case study.
Apparently, the complexity driving factor in the observed market is APP1. It is the reason behind a high outcome of the Complexity KPI. On a closer look, it turn out that the problem is embedded in the applications characteristics. Three out of four characteristics are in a poor state. However, the *number of covered capabilities* is difficult to reduce. One way may be to add another application in the portfolio which replaces APP1 in some of the capabilities. The downside are additional costs and one more application in the portfolio which increases the number of interfaces in the entire architecture. Which in turn results in increasing complexity of the overall EA. With connections to ten other applications the *number of interfaces* is relatively high. In order to reduce this, APP1 may be merged with some other applications in order to create one single application. However, this approach involves development effort and costs. As mentioned, APP1 is used by 8 business capabilities. But only three of them are in a critical state. One reason for this is that these business capabilities involve additional applications which are in a good condition, thus the overall complexity of these business capabilities is less bad. But in case of BC6, which also involves APP1, the complexity value of this application is low since it has interfaces to applications within the same capability. Consequently, by moving or adding the connected applications into the same business capability the value of this subfunction would be reduced by factor P, which is the penalty value for connections within the same capability. The *number of technology components* may be reduced. Based on the application's design this can be difficult and only achieved by replacing the application by another one with less technology components.
APP3 which creates numerous critical incidents is the main reason for the poor quality of BC4's application architecture. Furthermore, the processing time of the incidents is unusually high. Consequently, the number of incidents has to be reduced. If that is not possible at least the impact of the incident has to be decreased. This topic is usually addressed by the *Incident Management* which is also responsible for a acceptable processing time [TLR07].
The application architecture's critical state of failure impacts is primarily caused by APP1 and APP2. Here the main reason is that both applications are marked as business impact driving applications. This basically can be changed by increasing the redundancy. The more alternative applications exist which can perform the same functionality, the less is the failure of one of those applications. But one should be aware of potential consequences when adding additional applications into the portfolio: the complexity increases and additional costs arise.

6. Application Architecture Status Visualization

Back in 1854, John Snow, a physician located in London, created one of the first examples of data visualization. By using a map, which is illustrated in figure 6.1, he could identify the reason for the cholera outbreak in central London. He investigated the deaths and plotted their locations on a map. In addition, he overlaid the map with various water districts. As a result, he identified a pump located in a district with significantly many deaths. It turned out that this pump was the cause of the cholera epidemic. As this example shows, data visualization can be very helpful in identifying the cause of a problem or in the overall decision-making process, which makes it a part of this thesis.

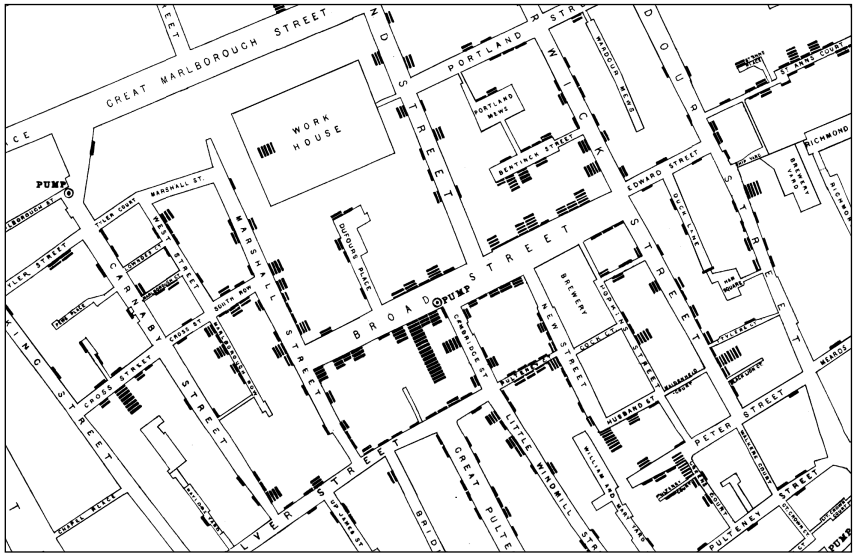


Figure 6.1.: Domain-driven design context map

This chapter describes the visualization of the application architecture status and includes the creation of a prototype. However, the general objective is to create an appropriate design which illustrates the in the previous chapter developed KPIs. Based on potential visualization types in EA (see section 6.1) and the imposed requirements by the industrial partner (see section 6.2.1) the prototype will be designed in section 6.2.2. The implementation of the prototype, which is not part of the thesis, is based on the data model presented in section 6.2.3.

6.1. Data Visualization Types in Enterprise Architecture

In order to create an appropriate visualization of the developed KPIs the state-of-the-art visualization types are analyzed. Therefore, a literature review on data visualization is conducted. As a result, a research paper of the sebis chair has been identified. The tool survey on EA visualization published by Roth (2014) compares a variety of EA tools on their visualization capabilities [Rot14]. One part of the paper describes general visualization types which are used in EA. Based on the opinion of EA practitioners, each visualization type was evaluated. According to Buckl's layered EA structure the aptitude for each layer has been analyzed. Following this thesis' objectives, only the application layer is considered as relevant. The table 6.1 illustrates the aptitude of each type, with a participation greater than 25.
Since the listed visualization types are well-known in the IT discipline, in the following

Visualization type	**Aptitude for application layer visualization**
ER diagram	66%
UML	61%
Matrix/Table	45%
Cluster map	35%
List	34%
Timeline	32%
Bar chart	30%
Tree view	30%
Flow diagram	27%
Graph	27%
Pie chart	27%
Radar chart	20%
Bubble chart	18%
Dashboard	14%
BPMN	12%

Table 6.1.: Relevance of visualization types for application layer $n > 25$ [Rot14]

only the well ranked ones will be described in detail. It appears from the results that an entity-relationship diagram (ER diagram) and the Unified Modeling Language (UML) are the most suitable visualization types for the application layer. The ER diagram is widely used to describe the attributes of an application as well as its relationship to other applications. Using different forms, the notation is unambiguous. The UML on the other side is another prominent visualization type. This modeling language offers multiple diagrams types, for example, class diagram, in order to illustrate the model structure. Although both visualization types primarily address the application layer, they are limited suitable for the purpose of KPI visualization. The ER diagram and UML are both providing very detailed information about applications. Consequently,

the stakeholders identified by the survey are mostly solution architects.
A matrix or a table are both more generic visualizations. Compared to the ER diagram or the UML, where the notation is strictly defined, a matrix can easily be combined with other visualization types. These specific illustrations can combine the benefits of several visualizations in order to satisfy particular information demands better than simple diagrams [Rot14]. Typically, a matrix is organized in rows and columns. According to the survey more than 75% of the participants use some kind of matrix or table. 45% of those use it to illustrate information related with the application layer. One more interesting fact is, that this visualization is mostly used by enterprise architects (34%), which makes a matrix/table to an eligible candidate for the prototype.
Cluster maps are often used to illustrate the hierarchical order of business domains and their relationships. The structure is visualized by nested rectangles. The outer rectangles usually represent the domain and the inner rectangles the involved applications or other IT components. In fact, this illustration is also used to show the structure of business capabilities. As shown in the foundation part, a business capability map is typically designed in such nested rectangles (figure 2.5). Since the second research question deals with the use of the business capability map, this visualization type seems to be very suitable.
Another very basic but also effective visualization type is a list. According to the survey conducted by Roth (2014) it is predominantly used to illustrate objects related to the business and application layer. In fact 34% of the participants use lists for application visualization. Usually lists consist of successive textual items separated by bars or bullet points. Although, this allows to communicate much information to the user it can be very unstructured. Hence the challenge when using lists is to maintain a clear structure to provide usability. However, lists are often preferred by enterprise architects [Rot14] and thus will be considered as a design element in the prototype.

6.2. Prototype Development

Within the scope of the thesis a prototype will be presented to visualize the developed KPIs. Resulting from the identified visualization types in the previous chapter and the requirements, which are presented in the next section, the design of the prototype will be established. Finally, based on the identified design and the data model a conceptual prototype will be presented.

6.2.1. Requirements Analysis

Requirements elicitation is a central part of software development. There are many techniques which can be applied in order to clarify the stakeholder's requirements. One of the most efficient way to gather this information is in form of interviews [Som07]. During the biweekly jour fixes with the industry partner some functional requirements for the prototype have been crystallized. Furthermore, a variety of non-functional requirements have been identified by conducting literature review.

Functional requirements

In the early stages of developing the prototype the functional requirements have been defined. The primary objective of this step is to avoid misunderstandings and to reduce the risk of complications in further stages. Put simply, functional requirements describe what a system should do [Som07]. It states how the system should react to particular inputs and situations. Over time the requirement have changed and new requirements emerged. In order to keep track the requirements are documented according to the *Volere requirements specification template* [RR12]. This template suggests to group similar requirements based on their purposes. To ensure the fulfillment of the requirements, a detailed description and a fit criterion is defined for each requirement. In the course of the conducted discussions four requirement types have emerged, which are illustrated in the following tables.

Requirement type	1
Requirement description	Feature requirements
Requirement	1.1
Name	PowerPoint export
Description	The system should provide a possibility to export the application architecture status to Microsoft PowerPoint.
Fit criterion	The system provides the option to generate a .pptx file of the application architecture status view.

Table 6.2.: Feature requirements

Requirement type	2
Requirement description	Interaction requirements
Requirement	2.1
Name	Dashboard
Description	The user is able to create an individual view beforehand.
Fit criterion	The system provides a dashboard with a form to select the required parameters.
Requirement	2.2
Name	Time period
Description	The user is able to select a individual time period.
Fit criterion	The system provides a possibility to select a certain time period.
Requirement	2.3
Name	Market
Description	The user is able to select a individual market.
Fit criterion	The system provides a possibility to select a certain market.
Requirement	2.4
Name	KPI type
Description	The user is able to select a required KPI.
Fit criterion	The system provides several KPIs the user can choose from.
Requirement	2.5
Name	Market comparability
Description	The user is able to compare the application architecture status of the selected market to other markets.
Fit criterion	The system provides a possibility to select a reference market, which ensures KPIs referring to the same market.

Table 6.3.: Interaction requirements

Requirement type	3
Requirement description	Capability visualization requirements
Requirement	3.1
Name	Layout
Description	The layout of the visualization should be familiar to the user.
Fit criterion	The layout of the visualization is based on the company's business capability map.
Requirement	3.2
Name	Easy to interpret
Description	The user has to be able to interpret the application architecture status easily.
Fit criterion	The application architecture status of the business capability and its applications is evaluated by a distinct colour scale.
Requirement	3.3
Name	High-level perspective
Description	The AL status is only relevant for level 1 capabilities.
Fit criterion	The system shows the AL status only for level 1 capabilities.
Requirement	3.4
Name	Strategic relevance
Description	Strategic relevant business capabilities should be highlighted.
Fit criterion	Strategic relevant business capabilities are highlighted by colouring.

Table 6.4.: Capability visualization requirements

Requirement type	4
Requirement description	Application landscape visualization requirements
Requirement	4.1
Name	Complexity
Description	The complexity of the application architecture should be visualized.
Fit criterion	The system provides a separate view which shows the Complexity KPI of the business capabilities.
Requirement	4.2
Name	Quality
Description	The quality of the application architecture should be visualized.
Fit criterion	The system provides a separate view which shows the Quality KPI of the business capabilities.
Requirement	4.3
Name	Impact
Description	The failure impact of the application architecture should be visualized.
Fit criterion	The system provides a separate view which shows the Impact KPI of the business capabilities.
Requirement	4.4
Name	Application view
Description	For each business capability the status of it's applications should be displayed.
Fit criterion	Displaying the application status by modals linked to the business capabilities.

Table 6.5.: Application visualization requirements

Non-Functional requirements

Non-functional requirements are not directly related to the features of the system. Rather than an individual service, they apply to the entire system as a whole [Som07]. Common non-functional requirements are listed in IS literature [CNYM12, RR12, Som07]. Based on discussions with the industrial partner the focus was on *organizational* and *usability requirements*. The usability requirements are fulfilled by using the 10 heuristics for user interface design by Nielsen (1994) [Nie94]. Since the objective of the prototype is merely to demonstrate a proof of concept, further non-functional requirements like security or performance are not taken into account.

Requirement type	5
Requirement description	Organizational requirements
Requirement	5.1
Name	Corporate layout
Description	The layout of the prototype should match with the predefined corporate standard.
Fit criterion	Use the corporate colours and fonts.

Table 6.6.: Organizational requirements

Requirement type	6
Requirement description	Usability requirements
Requirement	6.1
Name	Visibility of system status
Description	The system informs the user about current activity.
Fit criterion	The system displays information about the current view (e.g., selected date, market, KPI).
Requirement	6.2
Name	Match between system and the real world
Description	The user can understand the system's language.
Fit criterion	The system uses vocabulary familiar to the user.
Requirement	6.3
Name	User control and freedom
Description	The system provides a "emergency exit" in case the user selects a unwanted function.
Fit criterion	The system provides a "emergency exit" button in the upper left corner.
Requirement	6.4
Name	Consistency and standards
Description	The system's functions and interactions with the user should be consistent.
Fit criterion	The same layout and buttons are used for each KPI.
Requirement	6.5
Name	Error prevention
Description	The system should prevent errors effectively.
Fit criterion	No manual parameter input possible. No crashes in case of wrong parameter selection.

Table 6.7.: Usability requirements (part 1) [Nie94]

Requirement type	6
Requirement description	Usability requirements
Requirement	6.6
Name	Recognition rather than recall
Description	The user intuitively recognizes the interface.
Fit criterion	The layout of the visualization is based on the company's business capability map.
Requirement	6.7
Name	Flexibility and efficiency of use
Description	The system can be used efficiently by every user.
Fit criterion	The system provides a dashboard as a landing page, which allows the usage with no instructions needed.
Requirement	6.8
Name	Aesthetic and minimalist design
Description	The visualization should not contain irrelevant information.
Fit criterion	Only application or business capability related information is shown.
Requirement	6.9
Name	Help users recognize, diagnose, and recover from errors
Description	Error messages are expressed in a for the user understandable language.
Fit criterion	The system shows meaningful error messages.
Requirement	6.10
Name	Documentation and help
Description	The system should provide documentation and help. displayed.
Fit criterion	A legend describing the KPIs is provided on the landing page. A legend describing the colour scales is provided on each KPI page.

Table 6.8.: Usability requirements (part 2) [Nie94]

6.2.2. Prototypical Design

After analyzing the state of the art visualization types in EA and gathering the functional and non-functional requirements, the prototype can be designed. Based on the final design and the underlying data model the prototype is implemented. This process is illustrated in figure 6.2.

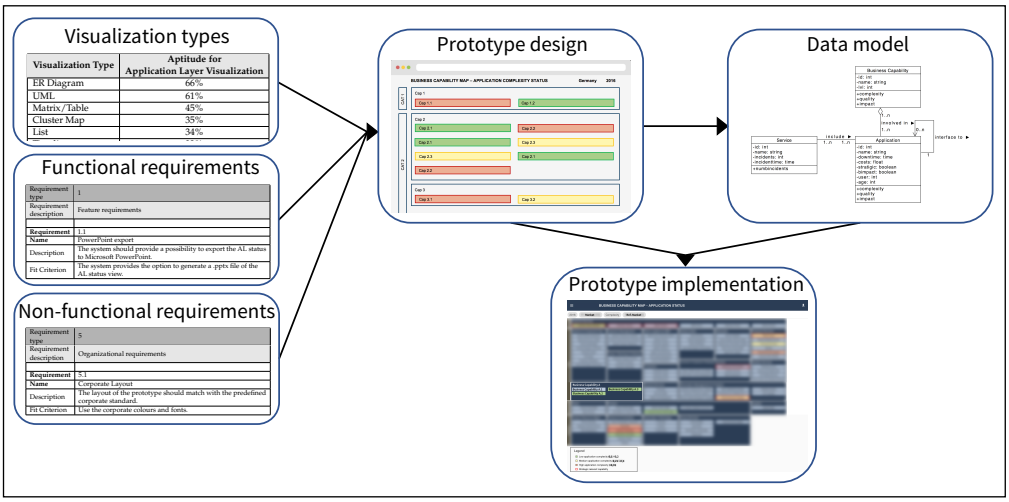


Figure 6.2.: Prototypical design process

Just like the development of the KPIs, the prototype is designed iteratively. Regular meetings and feedbacks from the industrial partners led to continuous improvements. For the design of the prototype attempts have been made to use the most suitable visualization types. The visualization types which are in line for that purpose are listed in table 6.1. It became early apparent that ER diagrams and UML are not the solution for visualizing an application architecture's status. However, the *cluster* map was established as the ideal visualization type. It is one of the most prominent ways to illustrate applications and it can be used to illustrate the business capabilities and the involved applications. Furthermore, a *list* become part of the visualization in order to illustrate the set of applications which support each capability. A *table* is included to provide further information about the applications. Here the rows describe each application characteristic and the column the value of the measurement. The overall logic behind the visualization is illustrated by figure 6.3.
Beside selecting and combining the right visualization types, the fulfillment of the functional and non-function requirements received highest priority. During the design process every single requirement was considered equivalently. As a result of iterative refinement, every single criterion could be met. In the following the most important requirements are described in detail.
Part of the *capability visualization requirement* was the provide a familiar layout. Thus, the organization's original business capability map was used as a foundation. To ensure easy and quick interpretation of the application architecture status another visualiza-

tion method was needed. For that reason, a literature review was conducted. In two scientific publications the use of colours was identified as an appropriate solution for that [UR11, Ben07]. Furthermore, only the level 1 capabilities were evaluated to provide a high-level perspective. Last but not least, the strategic relevant business capabilities were highlighted by a coloured border to attract additional attention.

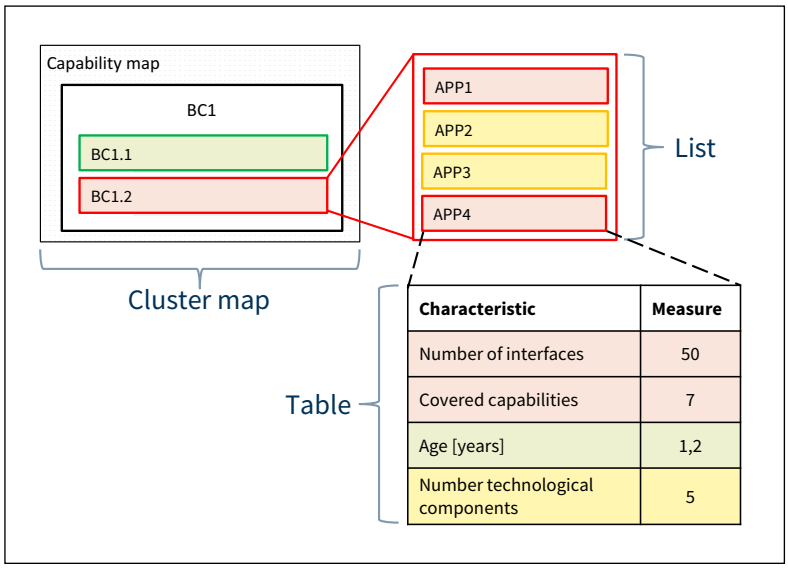


Figure 6.3.: Used visualization types

A major part of the *application architecture visualization requirements* was the distinct illustration of each KPI in different views. Thus, separate pages which show the application architecture's complexity, quality and impact are created. One important requirement was for the user to receive further information about each underlying applications. In order to meet this criterion, for each capability a list of the involved applications was added. Additionally, tables in form of modals show the involved application characteristics.
Part of the prototype design is to determine design elements to meet the *usability requirements*. In order to show the system status, the selected parameters are illustrated in the header of each KPI view. An exit button which brings the user back to the dashboard view is located in the upper left corner of the prototype. To reduce the error rate manual inputs are not possible at all. The user has to select the parameters via dropdown menus. In case of not selecting the required parameters, the prototype notifies the user. Legends are designed to support the user and to remove ambiguity. On the landing page, a legend describes involved application characteristics in each KPI. On each KPI view the ranges of the colour scales are illustrated in the lower left bottom.

6.2.3. Data Model

A data model basically displays a system's components and their relationships [Som07]. Put simply, it outlines the organization of a system, what makes it crucial for software development. Based on its prominence and widespread use, the data model was created in form of a UML class diagram, which is illustrated in figure 6.4. The attributes of

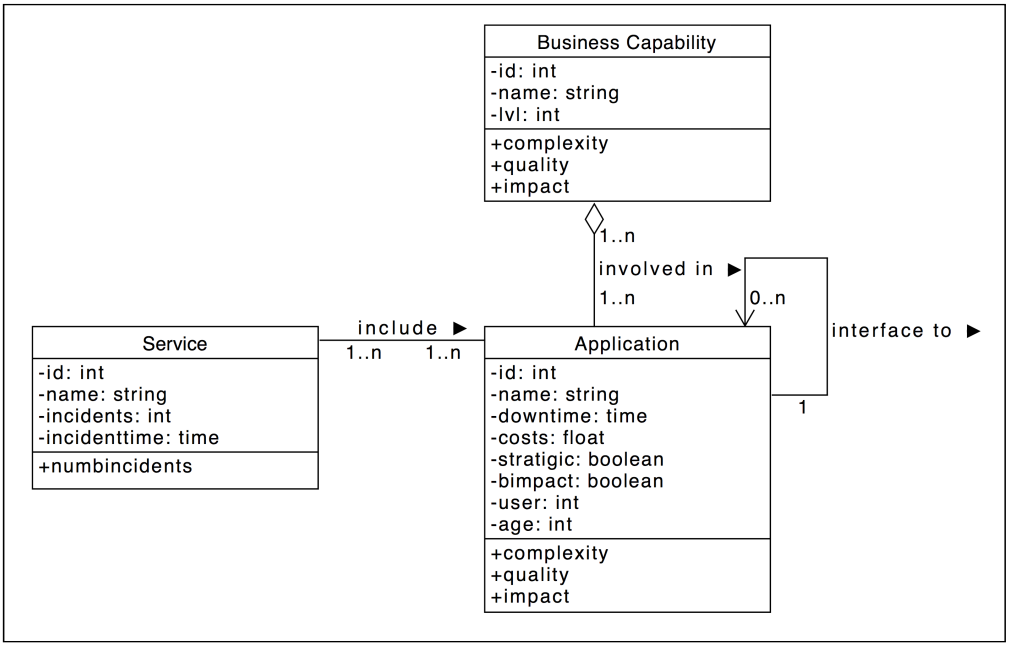


Figure 6.4.: Data model of the prototype

the application class represent the application characteristics and the KPI calculations are defined as operations. Since, according to, ITIL incidents are not directly mapped to applications but services there is a separated class where the incidents are added up. Via an association with the application class this information is aggregated to KPIs which address the business capabilities.

6.2.4. Prototypical Implementation

The real implementation of the prototype is not part of this thesis. However, a prototype was requested in order to evaluate the visualization design. In fact, the main reason for a prototype is to evaluate the proposals made by the developer for the design of the product [Smi91]. In this way misunderstandings and not considered requirements can be identified in an early stage.
The prototype is implemented by the sebis chair on Google's open-source front-end web application framework AngularJS which is one of the most used JavaScript frameworks [noe16]. Since the prototype is geared to the needs of the industrial partner it contains real data. As not to violate the secrecy agreement the screenshots of the prototype are censored. This however does not affect the traceability of the design. The landing page illustrated in figure A.1 satisfies the requirements, which include the selection of parameters and the legend. The figure 6.5 displays the view for the Complexity KPI. The colourings and the underlying range naturally differ for each view. A legend which shows the KPI range is displayed in the bottom left of the page. The PowerPoint export

Figure 6.5.: Application architecture complexity view

function, which is also a requirement is realized by the icon in the upper right corner of each view. By clicking it generates a .pptx file in fully adapted corporate design (see

figure A.6). Further information about an exemplary selected application is shown in the figures 6.6. The screenshots of the other KPI views are attached in the appendix.

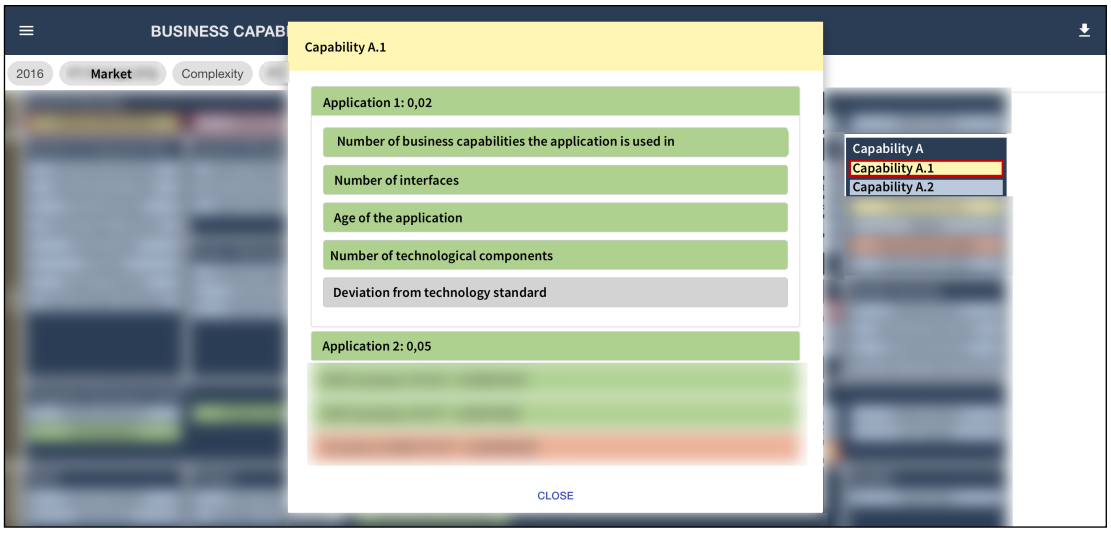


Figure 6.6.: Application complexity view

In the following chapter the approach and the results of the evaluation are described.

7. Evaluation of the Artifacts at a European Automotive Company

The evaluation is an essential part of design science approach according to Peffers [PTRC07] and Hevner [VAMPR04]. It basically measures the effectiveness and efficiency of the created artifacts and provides insights for improvements. In the following two sections the evaluation approach and the results are presented.

7.1. Evaluation Approach

The evaluation is conducted by expert interviews. For this purpose, meetings are arranged where the scope of the thesis was explained and the resulting KPIs and the prototype. Since the prototype includes real data provided by the industrial partner only employees of this company participated in the survey. People outside the company are not interviewed.
The survey is divided in three parts. Each part contains questions regarding the same topic. The first group of questions focuses on general information about the participant and his opinion towards APM. The next part contains questions evaluating the results of the KPIs and thus the status of the application architecture. The last questions are addressing the prototype. The primary objective here is to measure the usability based on the *Usefulness, Satisfaction, and Ease of use (USE) questionnaire* presented by Lund (2001) [Lun01]. An open field for comments is added. The complete questionnaire is attached in the appendix in section A.3.

7.2. Evaluation Results

Since APM and Business Capability Model address enterprise architects in the very first place, this group of stakeholer was targeted from the outset. Consequently, the first question shows that 6 out of 10 participants describe their job function as enterprise architect. The complete results of the first question is illustrated in figure 7.1. In order to ensure a sufficient level of expertise the second question addresses the number of years at work. The survey shows that 90% of the participants have 5 years or more professional experience (see figure A.7). Interesting are the results of the next two questions. Although, many share the opinion that the maturity level of EAM in the company is quite high, they dissent in the applied EAM Framework. However, the majority endorse the use of TOGAF. The next group of questions deal with the relevance of the selected KPIs. As shown in figure A.10, the participants are interested in every single

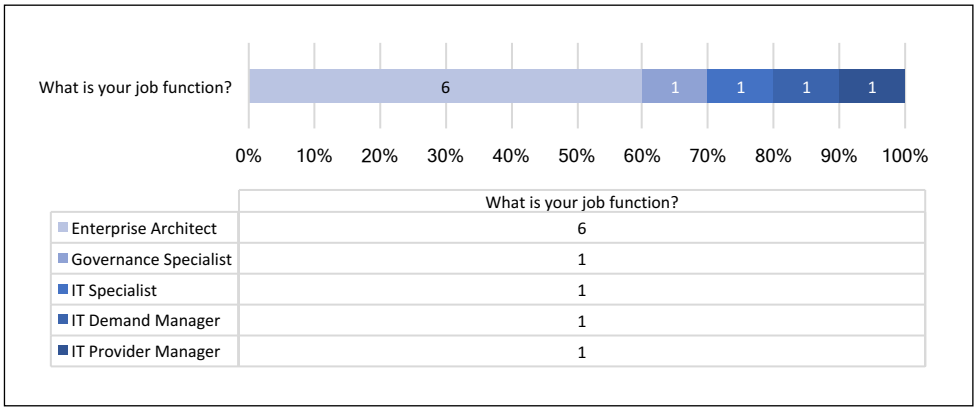


Figure 7.1.: Evaluation results: Job function

KPI. Nevertheless, there is a tendency of importance towards application architecture complexity. 80% of the participants consider the monitoring of complexity important. On the other side only 50% think the same for impact and 30% for quality. This supports the fact that complexity measurement attracts much attention in literature. The degree of expectation fulfillment regarding the KPI results was evaluated based on the outcomes of the case study. Thus, only participants familiar with the investigated market could make a statement. Although everyone agreed with the outcome, the survey shows that the results of the Complexity KPI are more accurate that the others. It reflects the reality slightly better than the Quality KPI but significantly better than the Impact KPI. As a consequence, this may mean that the Impact KPI has to be refined in the future. The last question group evaluates the usability and practicability of the prototype. Every single participant finds the tool intuitive, clear and easy to learn. This shows that especially the *usability requirements* are invariably fulfilled. Furthermore, the majority of the users share the opinion that this tool would reveal business capabilities with critical application architectures.
Since the evaluation was conducted in form of an interview the participants provided additional insights which are not captured by questionnaire. For the calculation of the current KPIs the applications are not weighted. One enterprise architect suggested that the applications should rather weighted based on their importance. Another participant suggested to rename the Impact KPI into *Effect KPI*. Nevertheless, most of the inputs are related to the visualization. In the early development stages a business capability map can change multiple times. New capabilities can be added, existing ones consolidated or removed. Thus, two participants want the map to be more flexible. Ideally, it should be adaptable easily by drag and drop each cluster. Furthermore, the stakeholder and the responsible person for each business capability should be displayed. Another participant is interested in an additional view which shows the status in an aggregated way. This can be realized by dividing each cluster into three parts each representing the status of one KPI. It was also required that based on the state a recommendation for action is displayed automatically. An enterprise architect with many

years of experience has seen the approach of using an individually programmed prototype as critical. Usually IT governance directives support the use of standard software instead. Despite the considerable number of suggestions and comments, the results of the evaluation are very promising. One of the most important insight resulting from this evaluation is the importance and high demand for application status monitoring. Especially in regard of complexity (see figure A.10).

8. Conclusion

This chapter summarizes the main finding in the research, which are presented in chapter 4 and 6. Additionally, the limitations of the thesis and suggestions for future work will be described in sections 8.2 and 8.3.

8.1. Summary and Conclusion

The chapters 1 and 2 represent the backbone of the thesis. The lack of linkage between APM and Business Capability Modelling was identified as a problem in literature. Based on that, research questions have been designed which are answered in the following chapters. The application characteristics are identified, filtered and categorized in chapter 4, which addresses the first research question. Existing APR methods were extended by Parmenter's KPI development procedure, which in turn was applied by using the Jolland's KPI aggregation method. As a result, a concept was presented to evaluate the status of application architecture of business capabilities. The continuously refined functions are used to calculate KPIs describing the complexity, quality and impact state. This part, answered also the **first research question** addressing the relevant application characteristics (see table 4.1). From the equations and the involved application characteristics recommendations for action are derived, which answer **research question three**. During the case study the importance of a consistent and complete database became evident. To gather the needed data, several systems and repositories were scanned. Consequently, a consistent database is one of the most important prerequisite for measuring and monitoring an application architecture. The second part of the thesis focuses on the proper visualization of the application architecture status. For this reason, a literature review on EAM visualization types has been conducted. The results of this research and the identified requirements built the foundation for the conceptual prototype design. The outcome was a business capability map which is composed of a cluster map, tables and lists. Thus, the **second research question** was answered by using the business capability map as the foundation of the visualization. Last but not least, the artifacts of the thesis are evaluated by expert interviews. The very promising results show that this topic attracts serious attention in the EAM and APM discipline.
From the recommendations for action it became clear that the improvement of each status is associated with costs. In many cases the existing applications should be replaced or extended to improve the application architecture's overall state. These actions are linked to significant costs. Therefore, for initial selection of applications it is essential to consider the presented application characteristics. For instance, choosing an application architecture which requires less interconnected applications would result is few

interfaces which in turn reduces the complexity. Furthermore, in some points compromises have to be made. Redundant applications for example reduce the possibility of serious impacts in case of failures but on the other side it increases the application architecture's complexity.
It is obvious that monitoring and measurement of application architecture states is beneficial in many ways. It alerts the organization in means of complexity, which leads to low agility. The quality of an application architecture indicates its failure-proneness. The impact view shows the consequences if such a failure occurs. Monitoring alerts the imminent catastrophes. However, they are also associated with costs. The reporting tool, which was presented in this thesis as a prototype, has to be developed properly. Additional manpower is needed for collection and cleansing of data. Hence it is essential to select only the most important KPIs to achieve your evaluation objectives consistent with your resources.

8.2. Limitation

During the development of the KPIs and design of the prototype following limitations crystallized out:

- The initial objective of this thesis was to create one centric KPI describing the general status of an application architecture. In the early stages it became clear that this would lead to an ambiguous statement. This was also supported by Jollands KPI framework, which says that subfunctions have to share the same point of interest. As a solution for this problem the KPI categories complexity, quality and impact were created. However, a statement about the overall application architecture status is missing.

- The list of identified application characteristics includes a tremendous amount of entries. This set was narrowed down based on expert opinion and the application data in hand. Therefore, it cannot be excluded, that only relevant application characteristics are not taken into account. For example, application size can still be an indicator for application architecture complexity.

- Following the terminologies of the business capability map presented by Ulrich and Rosen [Ros10] the prototype displays only the state of the second layer. The prototype design is unprovided for illustrating the application architecture state of level 3 business capabilities.

- Furthermore, each KPI refers to a distinct time period. In context of this thesis this period has been defined on an annual basis. This can easily be adapted into monthly periods by adjusting the data. Nevertheless, a hierarchical evaluation based on different time periods is not possible.

- As mentioned the data used for the case study was provided by the industrial partner. Since the data set involves highly sensitive data, a non-disclosure agree-

ment was signed. Consequently, only experts from the industrial partner were interviews during the evaluation. This may lead to a biased results.

8.3. Future Work

This thesis shows a promising concept for application architecture evaluation based on KPIs. Furthermore, a usable design was created which communicates the results of the evaluation to the user. Despite every effort, some open issues are identified while processing the thesis.

- **Further investigation on robustness**
  The developed KPIs are obviously based on mathematical equations. When creating a model it is essential to test the correlation of the subfunctions which are involved in the model. This has been done in section 5.1.2. However, the used data set is way to small for reliable results. Furthermore, the data set should be tested on outliers. Values which deviate from the mean significantly. Due to the size of the data set this was not taken into account. Nevertheless, these steps are important for robust KPIs and should be conducted on a sufficiently large data set.
  The weights included in the equations which define the relevance of each subfuncition are based on expert opinions. However, for contentual accuracy the weights have to be representative. For this purpose a survey with more experts would certainly increase the accuracy of the KPI outcome.

- **Additional visualization requirements**
  During the evaluation, further requirements regarding the visualization occurred. In this connection, an adaptable business capability map was requested multiple times. Since the capability map can change, its layout should be adaptable with little effort. By further refining the prototype these additional requirements can be met.

- **EAM tool connection**
  Currently, the data is stored externally and the KPIs are calculated in a separate tool. This results in manual action which has to be performed by an employee. The integration of the prototype into an EAM tools may allow automatic data transfer and KPI calculation within the tool.

Appendix

A. Appendix

A.1. Application Characteristics Correlations

```
*----------------*
*   COMPLEXITY   *
*----------------*

Correlation matrix
      int   cov   age  tech
int
cov  0.33
age  0.10 -0.19
tech 0.49  0.33 -0.05

p-values
      int  cov age tech
int
cov  0.10
age  0.62 0.36
tech 0.01 0.10 0.8

 *-------------*
 *   QUALITY   *
 *-------------*

Correlation matrix
             low   mid  high critical t_low t_mid t_high t_critical
low
mid         0.03
high        0.26  0.07
critical   -0.04  0.00  0.75
t_low      -0.02 -0.15  0.18     0.26
t_mid       0.00  0.17  0.01    -0.19 -0.04
t_high     -0.11 -0.14  0.03    -0.12  0.01  0.24
t_critical -0.30 -0.15 -0.17    -0.09 -0.03  0.00   0.82

p-values
             low  mid high critical t_low t_mid t_high t_critical
low
mid         0.90
high        0.19 0.73
critical    0.85 0.98 0.00
```

```
t_low      0.92 0.45 0.37     0.19
t_mid      0.99 0.38 0.97     0.35  0.86
t_high     0.59 0.48 0.87     0.57  0.98  0.23
t_critical 0.12 0.45 0.40     0.65  0.88  0.99      0
```

```
*------------*
*   IMPACT   *
*------------*

Correlation matrix
      cost    bi  user
cost
bi   -0.26
user -0.10 -0.33

p-values
     cost   bi user
cost
bi   0.25
user 0.64 0.13
```

A.2. Prototype Screenshots

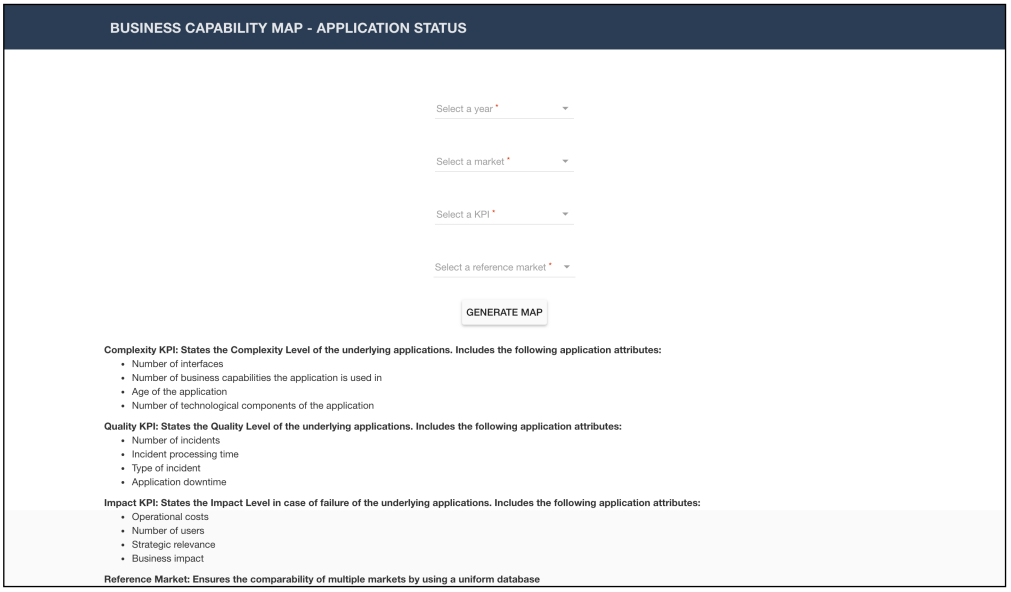


Figure A.1.: Landing page in form of a dashboard

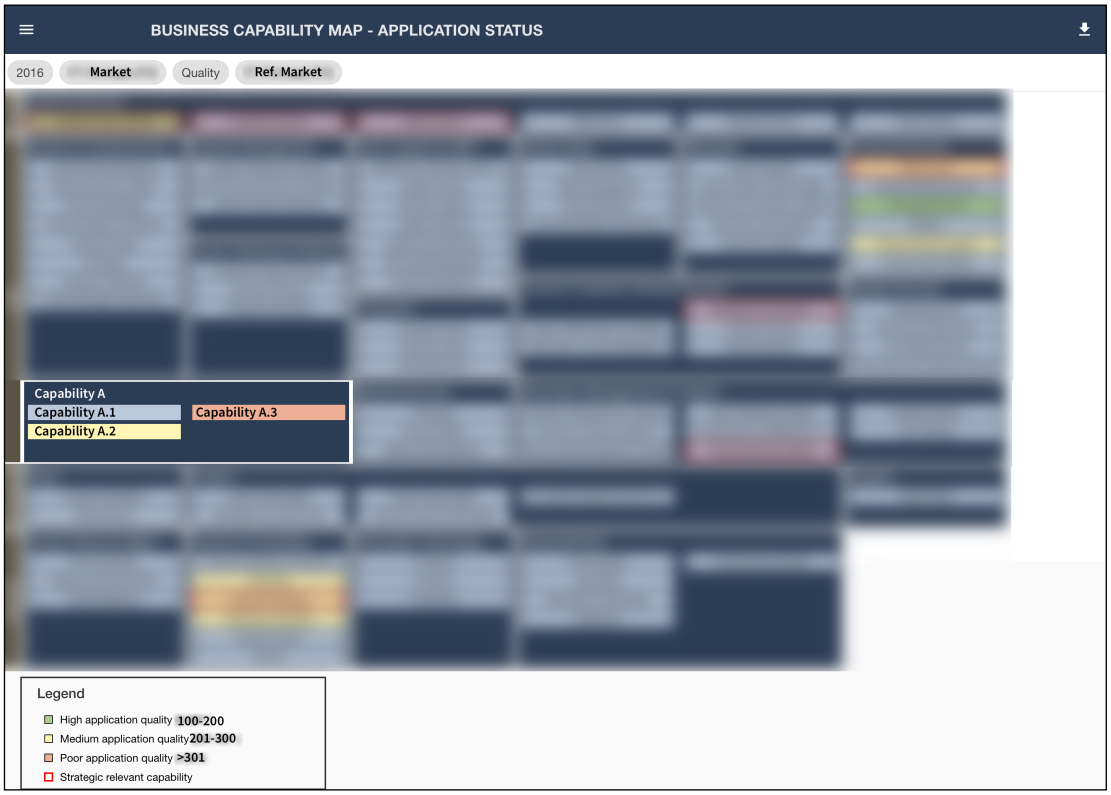


Figure A.2.: Application architecture quality view

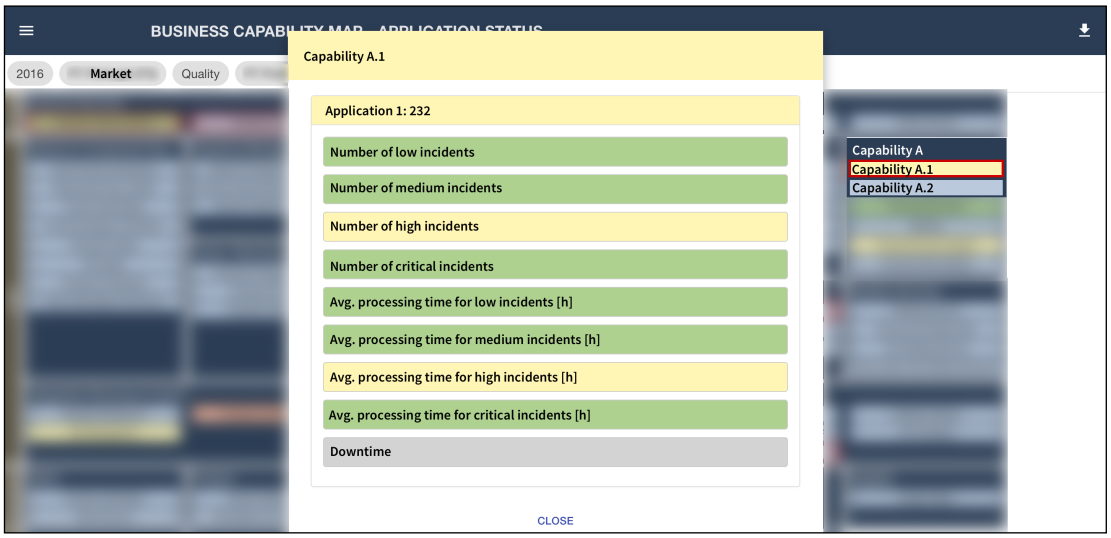


Figure A.3.: Application quality view

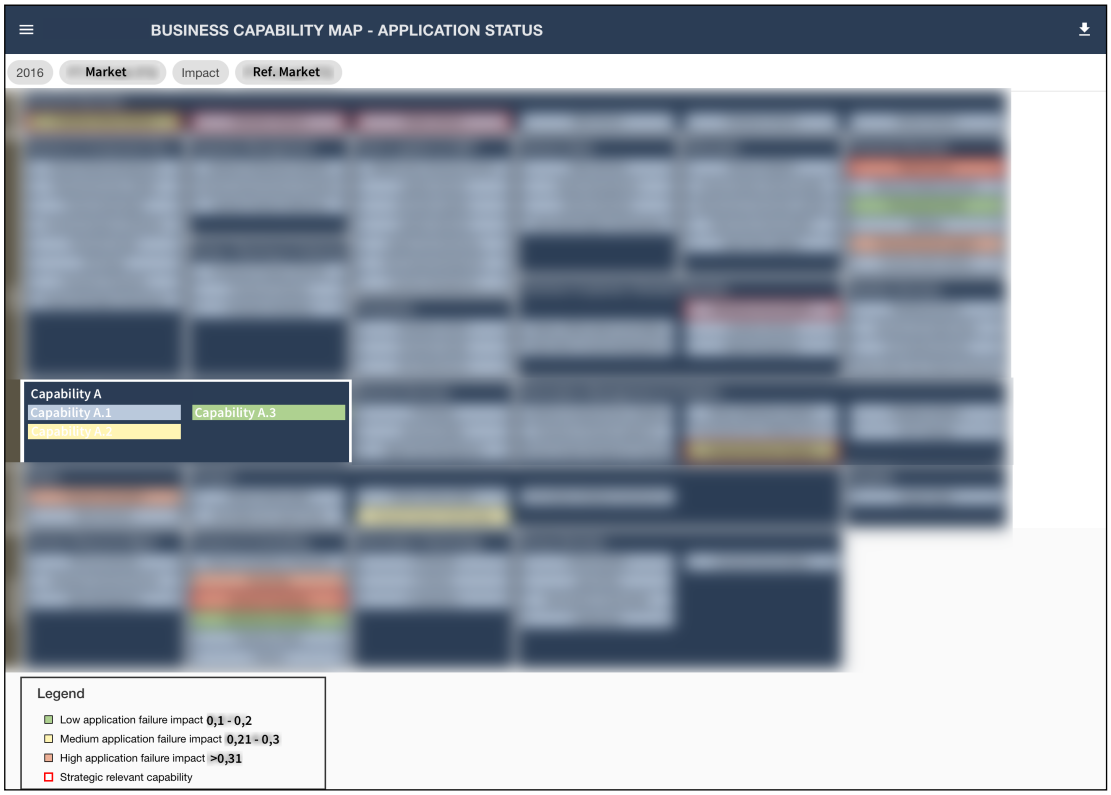


Figure A.4.: Application architecture failure impact view

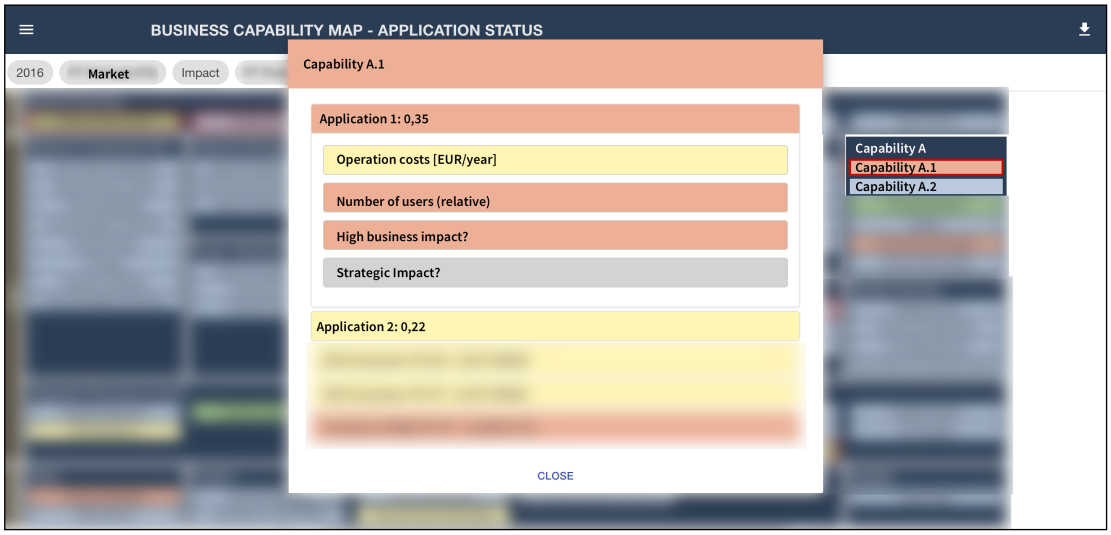


Figure A.5.: Application failure impact view

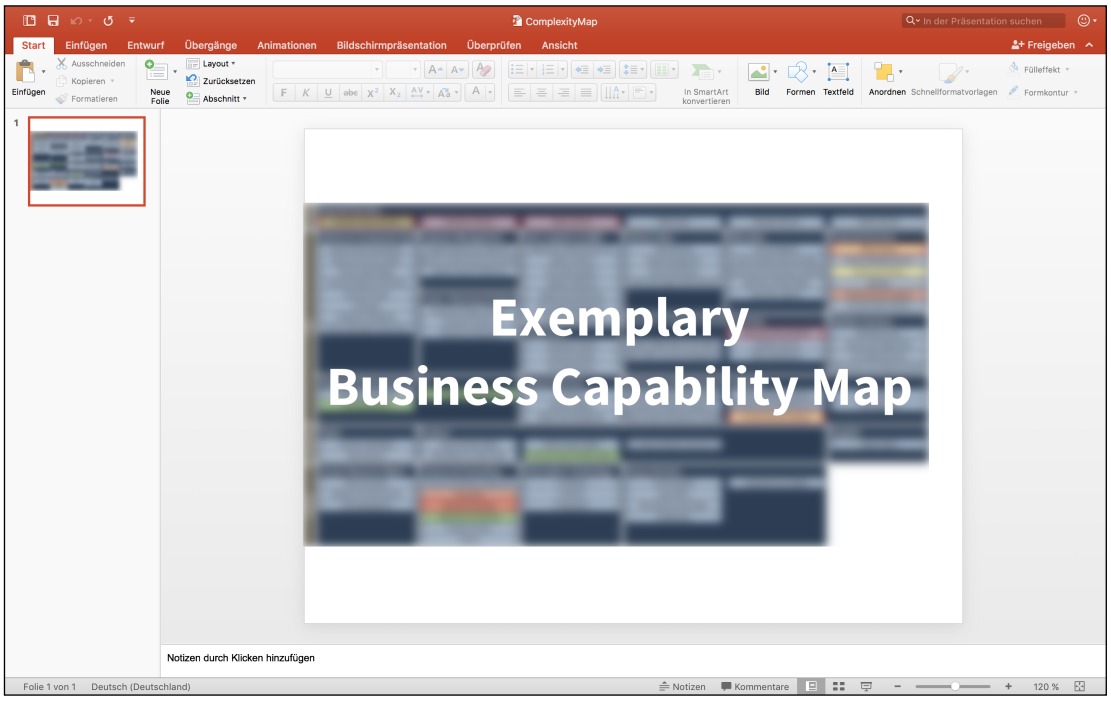


Figure A.6.: Exported Microsoft PowerPoint export

A.3. Evaluation Questionnaire

Technische Universität München

Prototype and KPI Evaluation

General questions:

1. **What is your job function?**

   _____

2. **How many years professional experience do you have?**

   _____

3. **What is the maturity level of the EAM initiative at your organization?**
   Please answer on a scale 0 to 5 - with 0 means that the level of maturity in your opinion is very low, and 5 stands for a very high level of EAM maturity. With values in-between you can adjust your statement.

Very low					**Very high**
☐	☐	☐	☐	☐	☐
0	1	2	3	4	5

4. **Which EAM framework do you use at your organization?**

Archimate	☐
IAF	☐
Quasar Enterprise	☐
TOGAF	☐
Zachman	☐
We do not use any framework	☐
Other	☐
Which one? _____	

5. **How important is the monitoring of the application landscape's complexity?**
   Exemplary factors: Number of interfaces...

Unimportant					**Very important**
☐	☐	☐	☐	☐	☐
0	1	2	3	4	5

6. **How important is the monitoring of the application landscape's quality?**
   Exemplary factors: Downtime, number of incidents,...

Unimportant					**Very important**
☐	☐	☐	☐	☐	☐
0	1	2	3	4	5

7. **How important is the monitoring of the impact in case of application landscape failure?**
   Exemplary factors: Number of affected users, business impact,...

Unimportant					**Very important**
☐	☐	☐	☐	☐	☐
0	1	2	3	4	5

Questions regarding the prototype:

8. **The tool control is very intuitive.**

I disagree					**I agree**
☐	☐	☐	☐	☐	☐
0	1	2	3	4	5

9. **The user interface design of the tool is clearly structured.**

I disagree					**I agree**
☐	☐	☐	☐	☐	☐
0	1	2	3	4	5

10. **The tool control is easy to learn.**

I disagree					**I agree**
☐	☐	☐	☐	☐	☐
0	1	2	3	4	5

11. **The tool provides transparency about issues in the applicaiton landscape and illustrates areas for action.**

I disagree					**I agree**
☐	☐	☐	☐	☐	☐
0	1	2	3	4	5

Questions regarding the KPIs:

12. **The presented complexity KPI meets my expectations.**

I disagree					**I agree**
☐	☐	☐	☐	☐	☐
0	1	2	3	4	5

13. **The presented quality KPI meets my expectations.**

I disagree					**I agree**
☐	☐	☐	☐	☐	☐
0	1	2	3	4	5

14. **The presented impact KPI meets my expectations.**

I disagree					**I agree**
☐	☐	☐	☐	☐	☐
0	1	2	3	4	5

15. Any further comments? (Pending points, extensions, changes, adjustments)

Thank you for your participation!

A.4. Evaluation Results

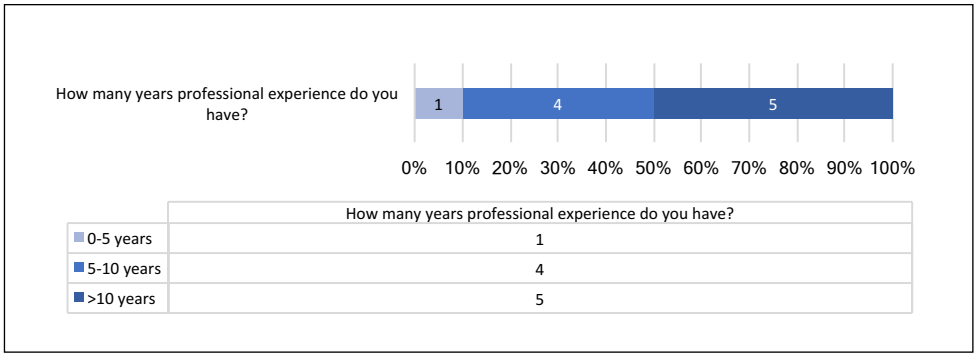


Figure A.7.: Evaluation results: Experience

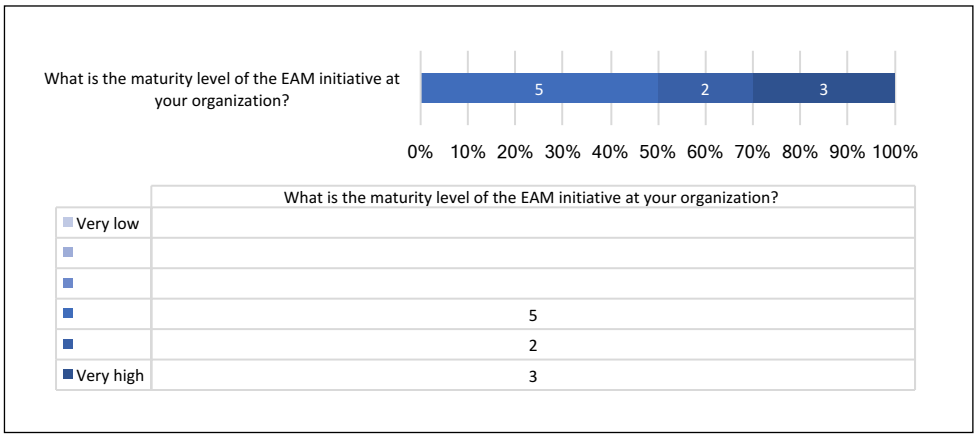


Figure A.8.: Evaluation results: EAM level

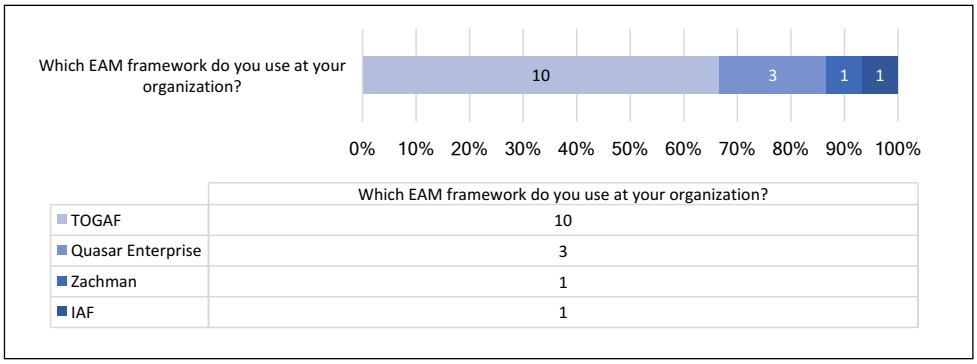
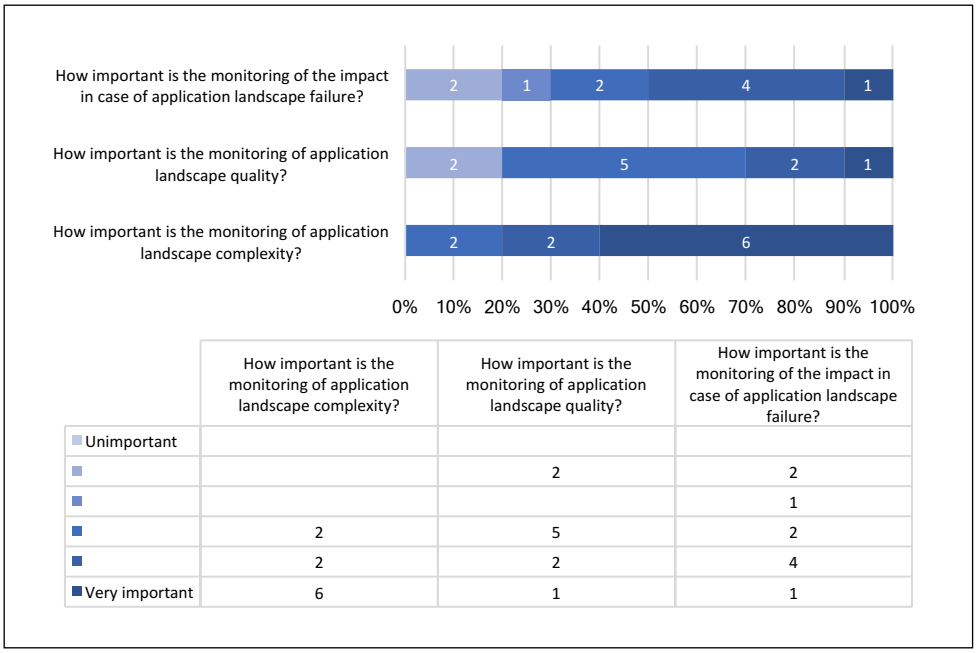
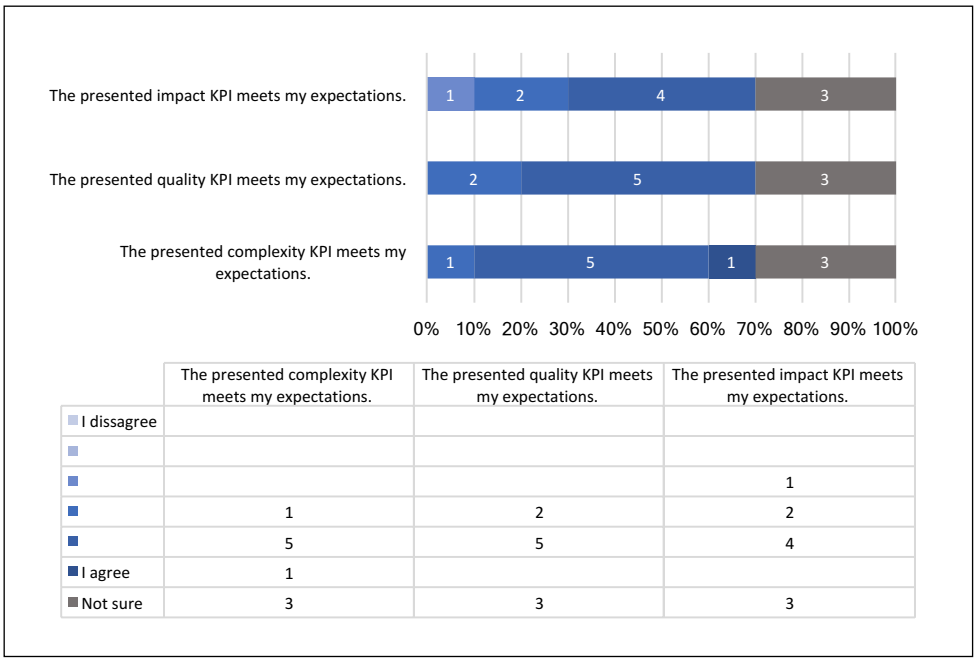


Figure A.9.: Evaluation results: EAM Framework



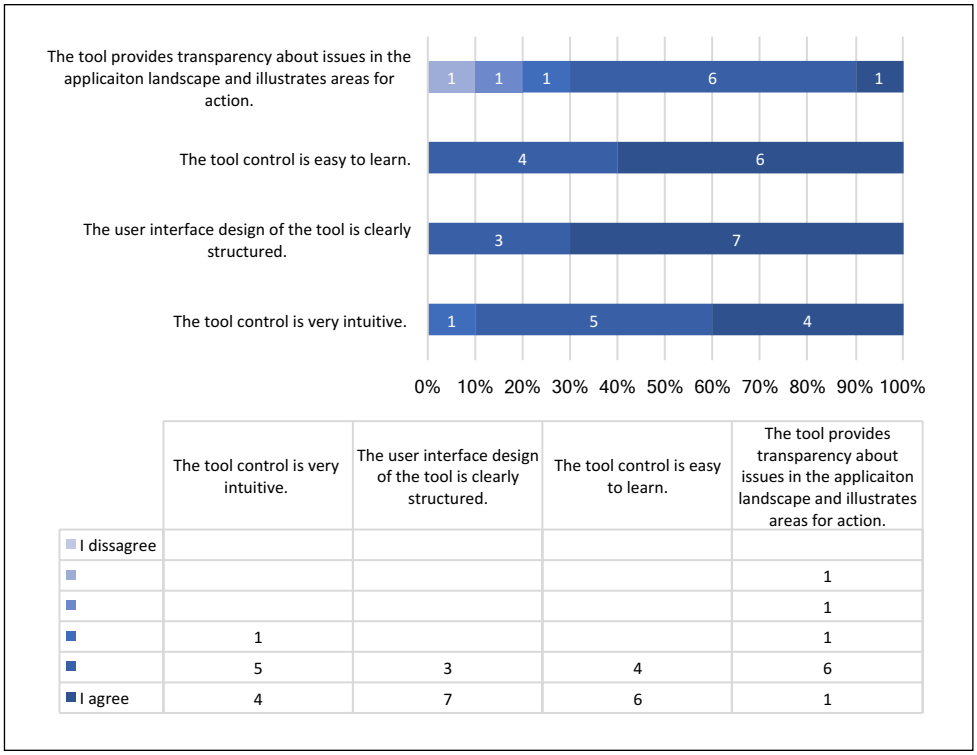
	How important is the monitoring of application landscape complexity?	How important is the monitoring of application landscape quality?	How important is the monitoring of the impact in case of application landscape failure?
Unimportant			
		2	2
			1
	2	5	2
	2	2	4
Very important	6	1	1

Figure A.10.: Evaluation results: KPI relevance



	The presented complexity KPI meets my expectations.	The presented quality KPI meets my expectations.	The presented impact KPI meets my expectations.
I dissagree			
			1
	1	2	2
	5	5	4
I agree	1		
Not sure	3	3	3

Figure A.11.: Evaluation results: Expectations fulfillment



	The tool control is very intuitive.	The user interface design of the tool is clearly structured.	The tool control is easy to learn.	The tool provides transparency about issues in the applicaiton landscape and illustrates areas for action.
I dissagree				
				1
				1
	1			1
	5	3	4	6
I agree	4	7	6	1

Figure A.12.: Evaluation results: Prototype

Bibliography

[AGW11] Stephan Aier, Bettina Gleichauf, and Robert Winter. Understanding enterprise architecture management design-an empirical analysis. 2011.

[AKBA16] Pouya Aleatrati Khosroshahi, Jannis Beese, and Stephan Aier. What drives application portfolio complexity? an empirical analysis of application portfolio cost drivers at a global automotive company. In *18th IEEE Conference on Business Informatics (CBI 2016), Paris*, 2016.

[AKBMW16] Pouya Aleatrati Khosroshahi, Jannis Beese, Florian Matthes, and Robert Winter. Causes and consequences of application portfolio complexity–an exploratory study. In *IFIP Working Conference on The Practice of Enterprise Modeling*, pages 11–25. Springer, 2016.

[AKHSM15] Pouya Aleatrati Khosroshahi, Matheus Hauder, Alexander Schneider, and Florian Matthes. Enterprise architecture management pattern catalog, version 2.0. *Technische Universität München, München*, 2015.

[ARW08] Ing Stephan Aier, Dipl-Wirtsch-Inf Christian Riege, and Robert Winter. Unternehmensarchitektur–literaturüberblick und stand der praxis. *Wirtschaftsinformatik*, 50(4):292–304, 2008.

[Ash07] Ron Ashkenas. Simplicity-minded management. *Harvard Business Review*, 85(12):101–109, 2007.

[Aus99] AusIndustry. Key performance indicators manual: A practical guide for the best practice development, implementation and use of kpis. *Business & Professional Publishing*, 1999.

[BAC11] Alexei Botchkarev, Peter Andru, and Raymond Chiong. A return on investment as a metric for evaluating information systems: taxonomy and application. *Interdisciplinary Journal of Information, Knowledge and Management*, 6:245–69, 2011.

[Bak00] Michael J Baker. Writing a literature review. *The Marketing Review*, 1(2):219–247, 2000.

[BBH07] J Brits, Gerrit Botha, and Marlien Herselman. Conceptual framework for modeling business capabilities. In *Proceedings of the 2007 informing science and IT education joint conference*, pages 151–170, 2007.

[BBK09] Günter Bamberg, Franz Baur, and Michael Krapp. Statistik, 15. auflage, 2009.

[BBL12] Stefan Bente, Uwe Bombosch, and Shailendra Langade. *Collaborative enterprise architecture: enriching EA with lean, agile, and enterprise 2.0 practices*. Newnes, 2012.

[Bee14] Karl Richard Beetz. *Wirkung von IT-Governance auf IT-Komplexität in Unternehmen: Beeinflussung der IT-Redundanz durch Verantwortungsteilung im IT-Projektportfoliomanagement*. Springer-Verlag, 2014.

[Ben07] Andreas Benson. *Qualitätssteigerung in komplexen Entwicklungsprojekten durch prozessbegleitende Kennzahlensysteme: Vorgehen zur Herleitung, Einführung und Anwendung*. Cuvillier Verlag, 2007.

[Ber15] Sofia Bergström. Modelling business capabilities withenterprise architecture: A case study at a swedish pension managing company. 2015.

[BMP10] Thiago Barroero, Gianmario Motta, and Giovanni Pignatelli. Business capabilities centric enterprise architecture. In *Enterprise architecture, integration and interoperability*, pages 32–43. Springer, 2010.

[BR96] Ulrike Baumöl and Thomas Reichmann. Kennzahlengestütztes ivcontrolling. *Zeitschrift für Controlling*, 8(4):204–211, 1996.

[Brä93] Annika Brändström. The stockholm jas fighter crash of 1993. 1993.

[Bro96] Martin Brogli. Steigerung der performance von informatikprozessen. *Vieweg, Braun*, 1996.

[Buc11] Sabine Buckl. *Developing Organization-Specific Enterprise Architecture Management Functions Using a Method Base*. PhD thesis, Technische Universität München, 2011.

[BVVD09] Eric Bouwers, Joost Visser, and Arie Van Deursen. Criteria for the evaluation of implemented architectures. Technical report, Delft University of Technology, Software Engineering Research Group, 2009.

[BW05] Christian Braun and Robert Winter. A comprehensive enterprise architecture metamodel and its implementation using a metamodeling platform. 2005.

[C+07] International Standardization Organization/International Electrotechnical Committee et al. Iso/iec 42010: 2007-systems and software engineering–recommended practice for architectural description of software-intensive systems. Technical report, Technical report, ISO, 2007.

[C+08] Joint Research Centre-European Commission et al. *Handbook on constructing composite indicators: Methodology and User guide*. OECD publishing, 2008.

[CK09] B Cameron and U Kalex. Webinar (web seminar) on business capability management. forrester research & alfabet ag, june 2009, 2009.

[CNYM12] Lawrence Chung, Brian A Nixon, Eric Yu, and John Mylopoulos. *Non-functional requirements in software engineering*, volume 5. Springer Science & Business Media, 2012.

[Cog86] F Coggeshall. The arithmetic, geometric, and harmonic means. *The Quarterly Journal of Economics*, 1(1):83–86, 1886.

[Coo07] Denise Cook. Business-capability mapping: Staying ahead of the joneses. *MSDN Library, URL:¡ http://msdn. microsoft. com/en-us/library/bb402954. aspx*, 2007.

[Cou99] CIO Council. Federal enterprise architecture framework version 1.1. *Retrieved from*, 80:3–1, 1999.

[DJ94] Gerardine DeSanctis and Brad M Jackson. Coordination of information technology management: Team based structures and computer based communication systems. *Journal of Management Information Systems*, 10(4):85–110, 1994.

[EHHJ08] Gregor Engels, Andreas Hess, Bernhard Humm, and Oliver Ju. *Quasar Enterprise: Anwendungslandschaften serviceorientiert gestalten*, volume 1. dpunkt Verlag, 2008.

[Eri10] Gunnar Eriksen. Managing business capabilities within strategic enterprise architecture. *Baseline*, (107):18–18, 2010.

[Eva04] Eric Evans. *Domain-driven design: tackling complexity in the heart of software*. Addison-Wesley Professional, 2004.

[FBvD07] Matthias Fabriek, Sjaak Brinkkemper, and Jeroen van Dullemen. A method for application portfolio rationalization. In *Digital Information Management, 2007. ICDIM'07. 2nd International Conference on*, volume 1, pages 466–472. IEEE, 2007.

[FHU07] Jürgen Fleischer, Markus Herm, and Jörg Ude. Business capabilities as configuration elements of value added networks. *Production Engineering*, 1(2):187–192, 2007.

[FMSN11] Andreas Freitag, Florian Matthes, Christopher Schulz, and Aneta Nowobilska. A method for business capability dependency analysis. In *International Conference on IT-enabled Innovation in Enterprise (ICITIE2011), Sofia*. Citeseer, 2011.

[Fre14] Andreas Freitag. *Applying Business Capabilities in a Corporate Buyer M&A Process*. Springer, 2014.

[FSH14] Florian Fittkau, Phil Stelzer, and Wilhelm Hasselbring. Live visualization of large software landscapes for ensuring architecture conformance. In *Proceedings of the 2014 European Conference on Software Architecture Workshops*, page 28. ACM, 2014.

[fSidI08] Bundesamt für Sicherheit in der Informationstechnik. *Kapitel 3: Business Impact Analyse*. , 2008.

[Gar10] Gartner. *Business Impact Analysis (BIA)*. http://www.gartner.com/it-glossary/bia-business-impact-analysis/, 2010.

[Gar11] Gartner. *Gartner Identifies 10 Key Actions to Reduce IT Infrastructure and Operations Costs by as Much as 25 Percent*. http://www.gartner.com/newsroom/id/1807615, 2011.

[Gar14] Gartner. *The Cost of Downtime*. http://blogs.gartner.com/andrew-lerner/2014/07/16/the-cost-of-downtime/, 2014.

[Gar15] Gartner. *Hype Cycle for Enterprise Architecture, 2015*. https://www.gartner.com/doc/3103125/hype-cycle-enterprise-architecture-, 2015.

[GBB12] Suzanne Gietema, Rik Bos, and Sjaak Brinkkemper. Reducing it related costs using application portfolio rationalization: A study focusing on reducing application portfolio size and complexity in small municipalities. In *ECEG2012-Proceedings of the 12th European Conference on e-Government: ECEG*, page 268. Academic Conferences Limited, 2012.

[GH13] Patrick M Georges and Josephine Hus. *Six figure management method: How to grow your business with the only 6 KPIs you'll ever need*. Kogan Page Publishers, 2013.

[GN74] Cyrus F Gibson and Richard L Nolan. *Managing the four stages of EDP growth*. Harvard Business Review, 1974.

[Gre09] Leonard Greski. Business capability modeling. *Architecture & Governance Magazine*, 5(7):1–11, 2009.

[Gro04] The Boston Consulting Group. *From IT Complexity to Commonality: Making Your Business More Nimble*, 2004.

[GSK05] Remco Groot, Martin Smits, and Halbe Kuipers. A method to redesign the is portfolios in large organisations. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, pages 223a–223a. IEEE, 2005.

[HT06] Ulrich Homann and Jon Tobey. From capabilities to services: Moving from a business architecture to an it implementation, 2006.

[Ins12] IT Governance Institute. *COBIT 5*, 2012.

[Int12] MEGA International. *4 Keys to Get a Simple, Organized View of your Application Landscape*, 2012.

[J+03] Nigel Jollands et al. The usefulness of aggregate indicators in policy making and evaluation: a discussion with application to eco-efficiency indicators in new zealand. 2003.

[JG07] Rowena Jacobs and Maria Goddard. How do performance indicators add up? an examination of composite indicators in public services. *Public Money and Management*, 27(2):103–110, 2007.

[Jon06] James V Jones. *Integrated Logistics Support Handbook (3: e ed.)*. McGraw-Hill, 2006.

[KA04] Dinesh Kumar and Babu J Alappat. Selection of the appropriate aggregation function for calculating leachate pollution index. *Practice Periodical of Hazardous, Toxic, and Radioactive Waste Management*, 8(4):253–264, 2004.

[KAV05] Stephen H Kaisler, Frank Armour, and Michael Valivullah. Enterprise architecting: Critical problems. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, pages 224b–224b. IEEE, 2005.

[Kel09] Wolfgang Keller. Using capabilities in enterprise architecture management. *White Paper, Object Architects*, 2009.

[Kel12] Wolfgang Keller. *IT-Unternehmensarchitektur: von der Geschäftsstrategie zur optimalen IT-Unterstützung*. dpunkt. verlag, 2012.

[KK07] Herbert Kargl and Martin Kütz. *IV-Controlling*. Walter de Gruyter, 2007.

[KLFK10] Christopher Klinkmüller, André Ludwig, Bogdan Franczyk, and Rolf Kluge. Visualising business capabilities in the context of business analysis. In *International Conference on Business Information Systems*, pages 242–253. Springer, 2010.

[KN96] Robert S Kaplan and David P Norton. *The balanced scorecard: translating strategy into action*. Harvard Business Press, 1996.

[Kro09] Per Kroll. *Simplify IT with Application Portfolio Management (APM) - Driving Application Rationalization and Consolidation*, 2009.

[KS03] Robert Kirchhof and D Specht. Ganzheitliches komplexitätsmanagement. *Grundlagen und Methodik des Umgangs mit Komplexität im Unternehmen, Wiesbaden: Deutscher Universitäts-Verlag*, 2003.

[KSS15] Svyatoslav Kotusev, Mohini Singh, and Ian Storey. Investigating the usage of enterprise architecture artifacts. In *ECIS 2015*, pages 1–12. Association for Information Systems (AIS), 2015.

[Küt11] Martin Kütz. *Kennzahlen in der IT: Werkzeuge für Controlling und Management*. dpunkt. verlag, 2011.

[Lan08] Josef K Lankes. *Metrics for Application Landscapes*. PhD thesis, Citeseer, 2008.

[Lan09] Marc Lankhorst. {Enterprise Architecture at Work: Modelling, Communication and Analysis (The Enterprise Engineering Series)}. 2009.

[LBMA14] Robert Lagerström, Carliss Baldwin, Alan MacCormack, and Stephan Aier. Visualizing and measuring enterprise application architecture: an exploratory telecom case. In *2014 47th Hawaii International Conference on System Sciences*, pages 3847–3856. IEEE, 2014.

[Lum16] Truth Lumor. Towards the design of an agile enterprise architecture management method. 2016.

[Lun01] Arnold M Lund. Measuring usability with the use questionnaire12. 2001.

[MA07] Floyd Marinescu and Abel Avram. *Domain-Driven Design Quickly*. Lulu. com, 2007.

[Mal09] Nick Malik. *Why Business Capabilities are not in the Zachman Framework*. https://blogs.msdn.microsoft.com/nickmalik/2009/08/13/why-business-capabilities-are-not-in-the-zachman-framework/, 2009.

[Mar52] Harry Markowitz. Portfolio selection. *The journal of finance*, 7(1):77–91, 1952.

[Mar12] Bernard Marr. *Key Performance Indicators (KPI): The 75 measures every manager needs to know*. Pearson UK, 2012.

[Mas06] Dieter Masak. *Moderne Enterprise Architekturen*. Springer-Verlag, 2006.

[Mat11] Dirk Matthes. *Enterprise Architecture Frameworks Kompendium: Über 50 Rahmenwerke für das IT-Management*. Springer-Verlag, 2011.

[MBLS08] Florian Matthes, Sabine Buckl, Jana Leitel, and Christian M Schweda. *Enterprise architecture management tool survey 2008*. Techn. Univ. München, 2008.

[MBM+97] Bedřich Moldan, Suzanne Billharz, Robyn Matravers, et al. Sustainability indicators. a report on the project on indicators of sustainable development. 1997.

[Mic08] Microsoft. *Business Architecture Resources (MSBA / Motion)*. https://blogs.msdn.microsoft.com/asehmi/2008/03/03/update-business-architecture-resources-msba-motion/, 2008.

[Mil15] Scott Millett. *Patterns, Principles and Practices of Domain-Driven Design*. John Wiley & Sons, 2015.

[Moc09] Martin Mocker. What is complex about 273 applications? untangling application architecture complexity in a case of european investment banking. In *System Sciences, 2009. HICSS'09. 42nd Hawaii International Conference on*, pages 1–14. IEEE, 2009.

[MST02] Maurizio Morisio, Ioannis Stamelos, and Alexis Tsoukias. A new method to evaluate software artifacts against predefined profiles. In *Proceedings of the 14th international conference on Software engineering and knowledge engineering*. ACM, 2002.

[Nie94] Jakob Nielsen. Heuristic evaluation. *Usability inspection methods*, 17(1):25–62, 1994.

[Nie06] Klaus D Niemann. *From enterprise architecture to IT governance*, volume 1. Springer, 2006.

[noe16] noeticsunil. *JavaScript Frameworks: The Best 10 for Modern Web Apps*. http://noeticforce.com/best-Javascript-frameworks-for-single-page-modern-web-applications, 2016.

[oD09] Department of Defense. *Capabilities-Based Assessment (CBA) User's Guide*, 2009.

[Ott78] Wayne R Ott. Environmental indices: theory and practice. 1978.

[Pap14] Anastasios Papazoglou. Capability-based planning with togaf® and archimate®. 2014.

[Par11] David Parmenter. *Key performance indicators: developing, implementing, and using winning KPIs*. John Wiley & Sons, 2011.

[Pod15] Daniel Podgórski. Measuring operational performance of osh management system–a demonstration of ahp-based selection of leading key performance indicators. *Safety science*, 73:146–166, 2015.

[PTRC07] Ken Peffers, Tuure Tuunanen, Marcus A Rothenberger, and Samir Chatterjee. A design science research methodology for information systems research. *Journal of management information systems*, 24(3):45–77, 2007.

[Ren15] Alexander von Rennenkampff. *Management von IT-Agilität: Entwicklung eines Kennzahlensystems zur Messung der Agilität von Anwendungslandschaften*. PhD thesis, Ilmenau, Techn. Univ., Diss., 2015, 2015.

[RGA07] Gerold Riempp and Stephan Gieffers-Ankel. Application portfolio management: a decision-oriented view of enterprise architecture. *Information Systems and E-Business Management*, 5(4):359–378, 2007.

[Ros10] M Rosen. Business processes start with capabilities. *Retrieve from http://www. bptrends. com/publicationfiles/12-07-10-COL-PM*, 20:26, 2010.

[Rot14] Sascha Roth. *Enterprise architecture visualization tool survey 2014*. epubli GmbH, 2014.

[RR12] S Roberson and J Robertson. Volere: Requirements specifcation template. *The Atlantic Systems Guild*, 2012.

[RWR06] Jeanne W Ross, Peter Weill, and David Robertson. *Enterprise architecture as strategy: Creating a foundation for business execution*. Harvard Business Press, 2006.

[S.A14] Capgemini S.A. *Application Landscape Report 2014*, 2014.

[Sar06] S Sarissamlis. A sea of applications: portfolio rationalization. *Nautilus Research*, 2006.

[Sch08] Marten Schöenherr. Towards a common terminology in the discipline of enterprise architecture. In *International Conference on Service-Oriented Computing*, pages 400–413. Springer, 2008.

[Sch16] Alexander Schneider. *Decision Support for Application Landscape Diversity Management*. PhD thesis, Technische Universität München, 2016.

[Ses07] Roger Sessions. Comparison of the top four enterprise architecture methodologies. 2007.

[SFS10] Daniel Simon, Kai Fischbach, and Detlef Schoder. Application portfolio management–an integrated framework and a software tool evaluation approach. *Communications of the Association for Information Systems*, 26(1):3, 2010.

[SM14] Alexander Schneider and Florian Matthes. Unternehmensarchitekturgestütztes controlling zur beherrschung der it-komplexität. *Controlling*, 26(12):694–699, 2014.

[Smi91] Michael F Smith. *Software prototyping: adoption, practice and management*. McGraw-Hill, Inc., 1991.

[Sok15] Berke Sokhan. *Domain Driven Design for Services Architecture*. https://www.thoughtworks.com/de/insights/blog/domain-driven-design-services-architecture, 2015.

[Som07] Ian Sommerville. Software engineering. international computer science series. *ed: Addison Wesley*, 2007.

[SRSM15] Alexander Schneider, Thomas Reschenhofer, Alexander Schütz, and Florian Matthes. Empirical results for application landscape complexity. In *System Sciences (HICSS), 2015 48th Hawaii International Conference on*, pages 4079–4088. IEEE, 2015.

[SS08] Najmus Saqib and Muhammad Tahir Siddiqi. Aggregation of safety performance indicators to higher-level indicators. *Reliability Engineering & System Safety*, 93(2):307–315, 2008.

[SSM16] Alexander Schneider, Christian Schweda, and Florian Matthes. Identifikation überalterter komponenten in einer it-architektur. 2016.

[Swa06] David S Sward. *Measuring the business value of information technology: practical strategies for IT business managers*. Intell Press, 2006.

[SWG13] Alexander Schütz, Thomas Widjaja, and Robert Gregory. Escape from winchester mansion–toward a set of design principles to master complexity in it architectures. 2013.

[SWK13] Alexander Schütz, Thomas Widjaja, and Jasmin Kaiser. Complexity in enterprise architectures conceptualiza-tion and introduction of a measure from a system theoretic perspective. *ECIS 2013 Proceedings*, pages 1–12, 2013.

[SWS+13] Christian Schmidt, Thomas Widjaja, Alexander Schütz, et al. Messung der komplexität von it-landschaften auf der basis von architekturmetamodellen: Ein generischer ansatz und dessen anwendung im rahmen der architektur-transformation. In *GI-Jahrestagung*, pages 1261–1275, 2013.

[SZ92] John F. Sowa and John A. Zachman. Extending and formalizing the framework for information systems architecture. *IBM systems journal*, 31(3):590–616, 1992.

[Tan05] Hüseyin Tanriverdi. Information technology relatedness, knowledge management capability, and performance of multibusiness firms. *MIS quarterly*, pages 311–334, 2005.

[TLR07] Sharon Taylor, Vernon Lloyd, and Colin Rudd. Itil version 3 service design: The office of government commerce. 2007.

[UR11] William Ulrich and Michael Rosen. The business capability map: the "rosetta stone" of business/it alignment. *Cutter Consortium, Enterprise Architecture*, 24(4), 2011.

[VAMPR04] R Hevner Von Alan, Salvatore T March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *MIS quarterly*, 28(1):75–105, 2004.

[VdL15] Marco Van der Linden. *Microservices Architecture Principle #3: Small Bounded Contexts Over One Comprehensive Model*. http://blog.xebia.com/microservices-architecture-principle-3-small-bounded-contexts-over-one-comprehensive-model/, 2015.

[VdV15] Kevin Van der Vlist. *Domain Driven Design*. Sogyo, 2015.

[vdZ96] Johannes Titus Maria van der Zee. In search of the value of information technology. Technical report, Tilburg University, School of Economics and Management, 1996.

[Ven08] Naray Venkataraman. Safety performance factor. *International Journal of Occupational Safety and Ergonomics*, 14(3):327–331, 2008.

[Ver09] TOGAF Version. 9, the open group architecture framework (togaf). *The Open Group*, 1, 2009.

[VST07] André Vasconcelos, Pedro Sousa, and José Tribolet. Information system architecture metrics: an enterprise engineering evaluation approach. *The Electronic Journal Information Systems Evaluation*, 10(1):91–122, 2007.

[Wal07] Mike Walker. Integration of enterprise architecture and application portfolio management. *Microsoft MSDN Enterprise Architecture Center*, 2007.

[WF06] Robert Winter and Ronny Fischer. Essential layers, artifacts, and dependencies of enterprise architecture. In *2006 10th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW'06)*, pages 30–30. IEEE, 2006.

[WJ03] George CS Wang and Chaman L Jain. *Regression analysis: modeling & forecasting*. Institute of Business Forec, 2003.

[WLR06] Harald Wesenberg, Einar Landre, and Harald Rønneberg. Using domain-driven design to evaluate commercial off-the-shelf software. In *Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications*, pages 824–829. ACM, 2006.

[WV99] Peter Weill and Michael Vitale. Assessing the health of an information systems applications portfolio: An example from process manufacturing. *MIS quarterly*, pages 601–624, 1999.

[WW02] Jane Webster and Richard T Watson. Analyzing the past to prepare for the future: Writing a literature review, 2002.

[Yin13] Robert K Yin. *Case study research: Design and methods*. Sage publications, 2013.

[Zac87] John A Zachman. A framework for information systems architecture. *IBM systems journal*, 26(3):276–292, 1987.