TECHNISCHE UNIVERSITÄT MÜNCHEN

Lehrstuhl für Informatik XIX

Decision Support for Application Landscape Diversity Management

Alexander W. Schneider

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Dr. Martin Bichler
Prüfer der Dissertation:	
1.	Univ.-Prof. Dr. Florian Matthes
2.	Univ.-Prof. Dr. Helmut Krcmar

Die Dissertation wurde am 29.10.2015 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 08.03.2016 angenommen.

Zusammenfassung

Viele Unternehmen verlassen sich zunehmend auf Informationstechnologien (IT) zur Abwicklung ihres Tagesgeschäfts. Heutzutage kann die Anwendungslandschaft größerer Unternehmen aus mehreren tausend Geschäftsanwendungen und mehreren zehntausend Schnittstellen bestehen. Unternehmensarchitekten, die für die Gestaltung und Weiterentwicklung solcher Anwendungslandschaften zuständig sind, sehen sich daher mit einer steigenden IT-induzierten Komplexität konfrontiert. Dennoch müssen sie täglich Gestaltungsentscheidungen treffen. Um Bauchentscheidungen zu verhindern, müssen die umfangreichen Architekturdaten aufbereitet und die Entscheidungsträger besser über den aktuellen Zustand sowie die Auswirkungen ihrer Entscheidungen informiert werden.

Diversität innerhalb einer Anwendungslandschaft ist einer der Haupttreiber für deren Komplexität. Deshalb besteht das Hauptforschungsziel dieser Arbeit darin, zu identifizieren, welche Art von Entscheidungsunterstützung Unternehmensarchitekten benötigen, um eine Anwendungslandschaft hinsichtlich ihrer Diversität zu gestalten. In der Literatur werden verschiedene quantitative Indikatoren zur Messung der Diversität von Anwendungslandschaften beschrieben. Diese basieren jedoch auf einem begrenzten konzeptionellen Verständnis von Diversität und bieten entsprechend kein ganzheitliches Bild für den Unternehmensarchitekten.

In dieser Arbeit wird ein konzeptuelles Rahmenwerk eingeführt, das auf Ideen und Erkenntnissen aus der Systemtheorie sowie der Ökologie fußt und ein ganzheitliches Verständnis von Diversität im Kontext einer Anwendungslandschaft ermöglicht. Basierend auf diesem Verständnis werden zwei neuartige Metriken vorgestellt, die die bislang unberücksichtigten Diversitätsaspekte messbar machen und somit die Entscheidungsunterstützung verbessern. Diese quantitative Entscheidungsunterstützung wird ergänzt durch ein System Dynamics Modell, das das typische Verhalten von Organisationen erklärt. In Verbindung mit einer Umfrage unter Praktikern zeigt dieses Modell Ansatzpunkte zur Einflussnahme für den Unternehmensarchitekten. Eine Reihe von vier Fallstudien beschreibt, wie die entwickelten Metriken zusammen mit ihrer Implementierung im Rahmen einer prototypischen Softwarelösung eingesetzt werden können, um Entscheidungen bezüglich der Gestaltung von Anwendungslandschaften in verschiedenen Unternehmen praktisch zu unterstützen. Durch die Verwendung der vorgestellten, neuartigen Indikatoren und des System Dynamics Modells können praktizierende Unternehmensarchitekten besser fundierte Entscheidungen bezüglich ihrer Anwendungslandschaft treffen. Außerdem können Forscher die Diversität von Anwendungslandschaften quantifizieren, um deren Auswirkung auf das Unternehmen zu untersuchen.

Abstract

Many organizations rely more and more on the usage of information technology (IT) to run their daily business. Today, the application landscapes of larger organizations typically consist of thousands of business applications and tens of thousands of interfaces. Enterprise architects responsible for designing and evolving such an application landscape are thus facing an increasing IT induced complexity. However, design decisions have to be made on a daily basis. To avoid gut decisions, the extensive architecture data needs to be processed to inform decision makers about the current state as well as impacts of their decisions.

Diversity within application landscapes is a major driver for their complexity. Therefore, the key research goal of this thesis is to identify what kind of decision support enterprise architects require to design an application landscape with regard to its diversity. In literature, several quantitative indicators measuring application landscape diversity are described, but all of them assume a limited conceptual understanding of diversity and thus do not provide a holistic picture to the enterprise architect.

This thesis introduces a conceptual framework, resting upon ideas and insights from system science and ecology, to describe the concept of application landscape diversity holistically. Based on this understanding, two novel indicators capable of quantifying diversity aspects are presented to aggregate architectural information. This quantitative decision support is complemented by a System Dynamics model explaining typical organizational behavior. Supplemented by a practitioner survey, this model provides suitable leverage points for exercising influence. A series of four case studies describes how these indicators as well as their implementation realized in a prototypical software solution, can be used to support application landscape design decisions at different companies in practice. By using the novel indicators and System Dynamics model presented in this thesis, practicing enterprise architects can make better informed design decisions regarding their application landscape. In addition, researchers are able to quantify diversity comprehensively to further assess its impact on the organization.

Acknowledgment

I would like to express my special appreciation and thanks to my supervisor Prof. Dr. Florian Matthes who gave me the freedom I needed and demanded results when required. I would like to thank him for encouraging my research and for allowing me to grow as a research scientist. I further want to express my sincere gratitude to Prof. Dr. Helmut Krcmar for being my second referee and for open an informal talks beyond research topics.

I would especially like to thank Prof. James Lapalme for offering me the opportunity to visit him and his research group and for substantially challenging my ideas. The time I spent with him was crucial for the development of this thesis. Likewise, I would like to thank Dr. Thomas Widjaja and Alexander Schütz for sharing their ideas on complexity. I also want to thank all participants of the CALM^3 workshop series as well as all case study partners for sharing their experience and worldview and supporting the whole research process. Without their help, it would have been difficult to focus on the relevant aspects of my research.

The sebis chair has provided an excellent environment for my research. Many ideas matured during hours of discussions with my colleagues and resulted often in co-authored papers. Therefore, special thanks go to Dr. Sabine Buckl, Dr. Christian Schweda, Dr. Ivan Monahov, Dr. Christopher Schulz, Dr. Christian Neubert, Dr. Alexander Steinhoff, Dr. Holger Wittges, Matheus Hauder, Marin Zec, Klym Shumaiev, Thomas Reschenhofer and Bernhard Waltl.

I would also like to thank the students who in one way or the other contributed to my research with their theses, guided research or as students working at sebis. Special thanks go to Anna Gschwendtner for her constant engagement in my research during her studies.

Last, but most of all, thanks to my family for their continuous support. I am most grateful to my parents for their support and encouragement during my whole life and my significant other Katharina Rau who gave me advice throughout my entire research endeavor and made sure that other important aspects in my life did not come off badly.

Garching b. München, 29.10.2015

Alexander Schneider

Table of Contents

List of Figures

List of Tables

List of Examples

CHAPTER 1

Motivation

"People are very open-minded about new things—as long as they're exactly like the old ones."

Charles Kettering

Today, many organizations are faced with changing environments and increasing uncertainty due to, e.g., rapid development of new technologies, increasing globalization of markets and the rise of innovative new forms of organization (Sanchez, 1997; Jones, 2010). Agility, i.e., the ability to detect and respond to opportunities and threats with ease, speed and dexterity, has emerged as a key business imperative (Tallon and Pinsonneault, 2011). To enhance their agility and transform their business, organizations rely more and more on Information Technology (IT) (Sambamurthy et al., 2003; Venkatraman, 1994). Thereby, organizations face the challenge of allocating resources for two different kinds of competing activities (March, 1991): exploratory activities and exploitative activities. Exploratory activities target long-term success by conducting research and experiments whereas exploitative activities target short-term success by establishing and refining routines (Levinthal and March, 1993). In other words, "exploitative innovations involve improvements in existing components and build on the existing technological trajectory, whereas exploratory innovation involves a shift to a different technological trajectory" (Benner and Tushman, 2002).

Consequently, conducting experiments to develop new, i.e., different, business models, business processes or products which are supported by IT requires continuous change of the organization's IT systems. For many years, companies focused their IT resources on delivering individual applications to address a specified business need and hosted each application on the best available technology platform (Ross et al., 2006). Although the resulting diversity among the IT systems

might provide benefits like less exposure to a single risk and shorter time-to-market of new business requirements, IT executives have been forced to focus on IT efficiency by continuously shrinking IT budgets. In 2014, a large study among companies headquartered in Germany, Austria or Switzerland revealed that 56% of these companies still have unchanged or even reduced IT budgets (Capgemini, 2014). In addition, several researchers argued that by eliminating diversity of IT systems, i.e., standardization, IT costs can be reduced significantly (Boh and Yellin, 2007). Therefore, many companies try to reduce the high diversity of IT-systems which resulted from uncontrolled IT evolution over many years.

One popular means to design and implement such standardization activities is to establish an Enterprise Architecture Management (EAM) function within the organization (Niemann, 2005, p. 135; Keller, 2012, p. 53; Hanschke, 2013, p. 98). The enterprise architects are commissioned to document the Enterprise Architecture (EA) which is defined as "the fundamental organization of a system [enterprise] embodied in its components, their relationships to each other and to the environment and the principles guiding its design and evolution" (International Organization for Standardization, 2007). Based on such architectural description which includes the various IT systems and their technical components, potentials for standardization can be identified. Supported by appropriate governance structures enterprise architects can heavily influence the degree of diversity among the organization's IT-systems. According to Ross et al. (2006) a standardized IT is the second level of EAM maturity after having business silos as a result of a period of uncontrolled growth.

Given the fact that IT standardization is associated with positive as well as negative effects (Ross et al., 2006; Boh and Yellin, 2007) taking place in a complex adaptive socio-technical system (Schneider and Somers, 2006; Sterman, 2000) a holistic understanding of IT diversity is required to efficiently and effectively design IT system landscapes. Especially in times of digital transformation (Andal-Ancion et al., 2003)—when IT systems are used to transform entire businesses or industries—balancing exploration and exploitation and respectively managing IT landscape diversity becomes essential. Up to now, the available theoretical and practical means supporting such decisions are limited in both, their conceptual understanding of diversity and their applicability. The following section further discusses the identified need for holistic IT system diversity understanding, details the problem statement and poses the research questions addressed by this thesis.

1.1. Problem statement and research questions

Given the need for organizational agility and therefore exploration activities which require diversity among IT systems as well as decreasing IT budgets and therefore exploitative activities resulting in standardized IT systems the task of appropriately designing an application landscape is not primarily a task of IT management alone. Instead, aligning business and IT is a goal best approached from both perspectives—business and IT (Orlikowski and Barley, 2001). The previously mentioned management function called EAM considers both, business- and IT-related concepts, but most preferably also accounts for cross-cutting aspects, such as strategies and projects. The latter is essential because managed evolution of the organization inevitably connects to the strategies as drivers of organizational change and projects as its vehicles (Murer

et al., 2008). Such holistic approach to the structured design of innovative and fundamental change is required to systematically support organizational transformation (Aier et al., 2009). Unfortunately, most traditional approaches to EAM focus only on the structure of an enterprise, i.e., its EA. Being negligent of dynamic aspects and involved human beings, current EAM approaches are often limited in their ability to support decisions regarding the diversity of an application landscape. According to that understanding, a preliminary definition of *application landscape* is provided to support an intuitive understanding of the term, which will be further discussed and refined in Chapter 2.2.

Preliminary definition: Application landscape
The application landscape of an enterprise covers the typical elements of an IT architecture including, for example, business applications, information flows, technology components and platforms which are used, developed, operated or managed by people.

The above definition emphasizes that an application landscape is part of a socio-technical complex adaptive system in which behavior is equally important as structure. Thereby, existing theories about how to manage an EA can be interrelated with complexity theory. As we know from socio-technical systems research (e.g. Emery (1959); De Greene (1973)) people and their used technologies should not be regarded separately which holds also true for enterprise architecting (Lapalme and de Guerre, 2013). Recent research on the complexity of systems found out that diversity within a system is one major driver for its complex patterns of behavior (Page, 2011). Therefore, the diversity within an application landscape should be of interest to each application landscape designer, because it affects the behavior of people and processes within an enterprise. According to Page (2011, p. 2), "even if on balance diversity's a good thing, we can have too much of it". Due to the fact that diversity can be regarded to be beneficial, the goal of an enterprise architect should be to maximize diversity within the application landscape. Nevertheless, there seems to be an upper limit of useful diversity. Thus, the enterprise architect can only introduce diversity to a given amount. Because it seems to be difficult to calculate such upper limit ex ante, the budget allocated for IT investments will be considered to form a natural boundary. Regarding this inherent challenge of finding the appropriate amount of diversity within a system, the main research question to be addressed in this thesis can be formulated as follows:

> *What kind of decision support do enterprise architects require to design an application landscape with regard to its diversity?*

Regarding the existing literature on complexity and diversity one can find many different understandings and interpretations of both terms (Johnson, 2007; Page, 2011). If enterprise architects should be enabled to manage application landscape diversity and make informed decisions, a clear understanding of diversity needs to be available. Up to now, the topic of IT standardization, i.e., the reduction of diversity, has drawn attention of many researches. However, most publications regarding this topic apply a limited view on diversity in the context of application landscapes although general considerations are much more elaborate. Accordingly, there is a gap between knowledge already generated by complexity scientists and knowledge currently used to examine application landscape diversity. By taking this mismatch into account, the first research question to be addressed in this thesis can be derived.

> ***Research question 1:*** Which aspects need to be considered during investigation of application landscape diversity?

Up to now, the essential role of diversity within complex systems has been identified, but to benefit from that insight, a more nuanced understanding of diversity in complex systems is required (Page, 2011). Because the concrete consequences of an increase or decrease of diversity within a system are very specific, a deeper understanding of impacts of standardization or diversification activities within an application landscape needs to be developed. This includes positive as well as negative effects and also reasons why companies today try to standardize or diversify their application landscape. Accordingly, the second research question of this thesis can be derived.

> ***Research question 2:*** How do changes of diversity within an application landscape affect its enterprise in general and the application landscape in particular?

For many business executives measuring is a prerequisite to management. For example, according to Harrington (1999) "Measurement is the first step that leads to control and eventually to improvement. If you can't measure something, you can't understand it. If you can't understand it, you can't control it. If you can't control it, you can't improve it". Despite some doubts about the benefit of business measurements in general, for example the adjustment of behavior towards metric improvement not firm improvement (Spitzer, 2007), using metrics facilitates learning and might prevent one from making the same mistakes again, especially in the software engineering domain (DeMarco, 1982). Based on this assumption, the third research question can be derived.

> ***Research question 3:*** Which indicators are suitable to quantify diversity aspects within application landscapes?

Assuming that suitable indicators for application landscape diversity quantification can be identified, actually calculating such indicators can be time-consuming. EAs typically consist of many elements impeding manual indicator calculations. Therefore, the need for a tool supporting diversity indicator calculations arises. Beside providing performant calculations, such tool needs to be able to handle organization-specific EA models and is ideally integrated with an existing EAM tool. This leads to the fourth research question of this thesis.

> ***Research question 4:*** How could a software architecture and respective implementation of a tool supporting application landscape diversity indicator calculations look like?

Although enterprises form complex systems in which emergent phenomena can be observed (see Schneider and Somers, 2006), they are also designed systems in the sense that, for example, the Chief Executive Officer (CEO) in general has the mandate to reconfigure all elements of an enterprise. If enterprise architects should be enabled to manage application landscape diversity they need to be aware of available tools or mechanisms they can use to influence it. The urge for knowledge about possible control mechanisms leads to the fifth research question.

> ***Research question 5:*** By which means can enterprise architects influence the diversity of application landscapes?

As one can see, the five research questions derived above can be distinguished by pursuing either an epistemic goal or a design goal (see Becker, 1995, p. 133). Epistemic goals pursue understanding of reality. Therefore, research questions 1, 2 and 5 fall into this category. Together they pursue the goal of developing a deeper understanding of the concept *diversity* in the context of application landscapes. By contrast, design goals are goals targeting a transformation of reality. This holds true for research questions 3 and 4. Together, they pursue the goal of designing tool-based metric support for decisions regarding the diversity of application landscapes.

By achieving these design and epistemic goals, this thesis equally addresses academics and practitioners facing the challenge of designing or analyzing the diversity of application landscapes. For practitioners the benefits include:

- The provision of concrete indicators capable of quantifying diversity within application landscapes including respective software support which can be used to support application landscape design decisions.
- The provision of a list of typical consequences of application landscape diversity which can be used to assess the adequacy of current application landscapes.

For academics concerned with diversity analysis of application landscapes this thesis provides the following results:

- A discussion and positioning of various approaches to understand and measure application landscape diversity.
- A better assessment of theoretical artifacts through the provision of several case studies.

1.2. Contributions of this thesis

The objective of this thesis is to contribute to the knowledge base regarding the design of application landscapes with respect to their diversity. Therefore, decision support will be provided in the form of a conceptual application landscape diversity framework operationalized through concrete metrics and tool-support. The contributions of this thesis include:

C1: A conceptual **complexity framework** to distinguish different notions of complexity

C2: An overview about the different **consequences of diversity** within application landscapes

C3: A conceptual **diversity framework** to distinguish between different aspects contributing to diversity

C4: A set of **indicators** for application landscape diversity quantification

C5: A prototypical implementation of an integrated **software solution** for application landscape diversity calculations

C6: An overview about possible **mechanisms to control or influence application landscape diversity** based on the complex system paradigm

The conceptual **complexity framework** consists of four different dimensions organizing eight

different notions of complexity which can be frequently found in literature. Thereby, a classification of the existing literature especially within the EAM discipline is possible and reveals pervasive and underrepresented complexity notions. The framework is complemented by a simple notation to clearly explicate applied notions of complexity which can be used to avoid misunderstandings. Both, the framework and its notation, might help to structure discussions and finally establish a shared understanding of complexity within the EAM domain. It serves as a basis for forthcoming diversity considerations (see Chapter 2.4.1).

The overview about different **consequences of diversity** within application landscapes provides a better understanding of application landscape diversity, thus setting the stage for all upcoming considerations. Thereby, positive as well as negative effects will be regarded equally. Furthermore, it will be shown how application landscape diversity contributes to the perceived complexity of application landscapes. Based on the identified consequences organization-specific diversity goals and strategies can be derived (see Chapter 3).

The conceptual **diversity framework** is required to explicate the term diversity within the context of application landscapes. It unifies previous research on diversity from a complex systems as well as ecological point of view. The framework provides a deeper understanding of application landscape diversity by clearly distinguishing between diversity within a type and diversity between types. Thereby, it relies on Description Logics (DLs) and organization-specific EA meta-models. Examples from industry will demonstrate how the conceptual diversity framework can be mapped to EA meta-models currently used in practice (see Chapter 4).

Based on the diversity framework, a **set of indicators** quantifying application landscape diversity is developed. Therefore, existing indicators found in practice or in EAM literature are analyzed and complemented, for example, by an adapted indicator typically used within the ecology discipline to assess species diversity. By applying all identified indicators to four real-world use cases the expressiveness of these indicators is demonstrated. Finally, their utility in terms of benefits and drawbacks is assessed by interviewing industry experts (see Chapter 5).

The prototypical **software solution** presented in this thesis provides an implementation of the identified application landscape diversity indicators. The prototype consists of a calculation component developed within a state-of-the-art open source computing environment. Metric implementations are provided as a library which can be used by scientists as well as practitioners. To demonstrate its integrability with existing EAM tools the calculation component is integrated seamlessly with the HybridWiki implementation Tricia (see Matthes et al., 2011). The resulting software solution satisfies many common EAM tool requirements (see Chapter 5.5).

To support enterprise architects in practice and to provide a foundation for future research an overview of currently known **mechanisms to control or influence application landscape diversity** is presented. Such mechanisms are primarily identified via a practitioners survey. Based on the concept of complex dynamic systems (see Sterman, 2000) a System Dynamics (SD) model is developed to show typically existing feedback loops within the application landscape system influencing its diversity. By taking several perspectives of different application landscape design stakeholders into account, such SD model seems to be able to foster shared understanding among application landscape designers thus enhancing design decisions (see Chapter 6).

While the conceptual frameworks and overviews about diversity consequences and control mech-

anisms provide mainly theoretical contributions, the novel indicators and prototypical tool support follow the design paradigm. To illustrate their usage and demonstrate their applicability four comprehensive **case studies** are conducted within different industry sectors and companies of varying size (see Chapter 7).

1.3. Epistemological position and research design

The research object of this thesis, i.e., decision support regarding application landscape diversity design, qualifies for the domain of Information Systems Research (ISR) which is focusing on Information System (IS) in organizations. According to Frank (2006), four major schools of thought exist within the philosophy of science which are relevant for ISR: logical positivism (see Carnap, 2003), critical rationalism (see Popper and Kiesewetter, 2003), critical theory (see Habermas, 1981) and the Erlangen constructivism (see Lorenzen, 1974). As Frank (2006) points out none of these schools satisfies all requirements of ISR. Nevertheless, within ISR a plethora of research methods is accepted (Wilde and Hess, 2007) allowing for both behavioral and design science research (Vessey et al., 2002). For this research endeavor the research goals are motivated from practice and will therefore be approached together with industry representatives if appropriate. The goal of this thesis is to develop artifacts like models, methods or metrics which implies the application of the design science paradigm (March and Smith, 1995; Hevner et al., 2004). According to Frank (2006), design science research builds upon two important assumptions. First, the design of artifacts can be a sophisticated task that contributes to the development of new knowledge on a scientific level. Second, the scientific design of artifacts is supposed to require a specific research method. Hence, the goal of the design science paradigm is utility (Hevner et al., 2004) which is achieved by developing innovative artifacts to solve relevant problems by applying rigorous research methods. Therefore, it can be differentiated from the behavioral science paradigm whose goal is truth (Hevner et al., 2004), achieved by developing and validating theories (see Figure 1.1).

The ISR research framework depicted in Figure 1.1 integrates the complementing research paradigms and consists of the following additional elements (Hevner et al., 2004):

- *Environment.* The environment defines the problem space in which reside the phenomena of interest. For ISR, it is composed of people, organizations, and their technologies.
- *Business Needs.* The business needs are defined by the goals, tasks, problems, and opportunities as they are perceived by people within the organization. They form the basis of ISR initiatives and can be assessed and evaluated within the context of organizational strategies, structure, culture, and existing business processes.
- *IS Research.* ISR is conducted in two complementary phases. Behavioral science develops and justifies theories that explain or predict phenomena related to the identified business need. Design science builds and evaluates artifacts designed to meet the identified business need.
- *Knowledge Base.* The knowledge base provides the raw materials from and through which

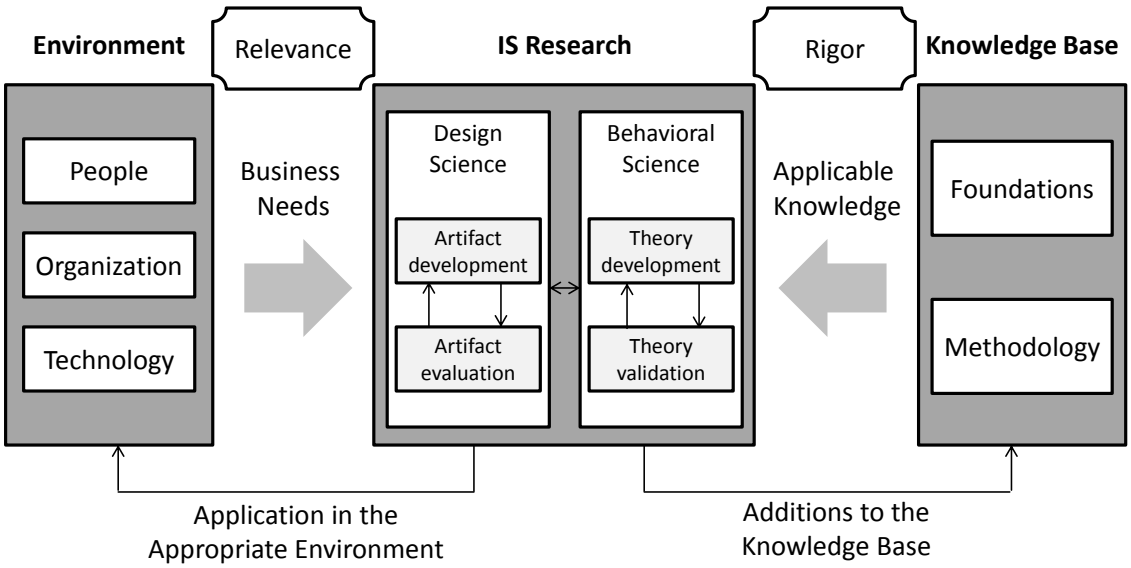


Figure 1.1.: Information systems research framework (Hevner et al., 2004)

ISR is accomplished. It is composed of foundations and methodologies, e.g. theories, methods and models.

- *Applicable Knowledge.* During a concrete research endeavor the knowledge base is available as applicable knowledge to the scientist.

The relevance of the problem addressed in this thesis is achieved by its motivation from practice and a strict alignment to actual business needs. Likewise, rigor is achieved by consequently taking the available knowledge base into account and following accepted research methods during artifact building. In general, this thesis will follow the research approach proposed by Peffers et al. (2007) which provides a nominal process model for doing design science research and serves as a mental model for presenting and evaluating design science research activities within the IS domain. The applied process follows a problem-centered initiation and is instantiated as visualized in Figure 1.2.

As a first step within the research process visualized in Figure 1.2, the actual problem is defined. As already outlined in Chapter 1.1, the problem addressed by this thesis is that there is no satisfying decision support available for enterprise architects regarding decisions affecting the diversity within an application landscape. Nevertheless, there is a strong need for such decision support because the effects of high or low diversity have impact on the overall firm performance. Furthermore, application landscape diversity seems to be crucial for innovations but also increases IT costs. To substantiate this motivation, practitioners are surveyed to identify the need and value of decision support regarding application landscape diversity design (see Chapter 3). Without appropriate decision support enterprise architects have to rely on their gut feeling which might result in suboptimal solutions.

Within the next step, the objectives of a solution are defined. Therefore, a detailed review of existing aspects contributing to diversity forms the basis for the development of the conceptual

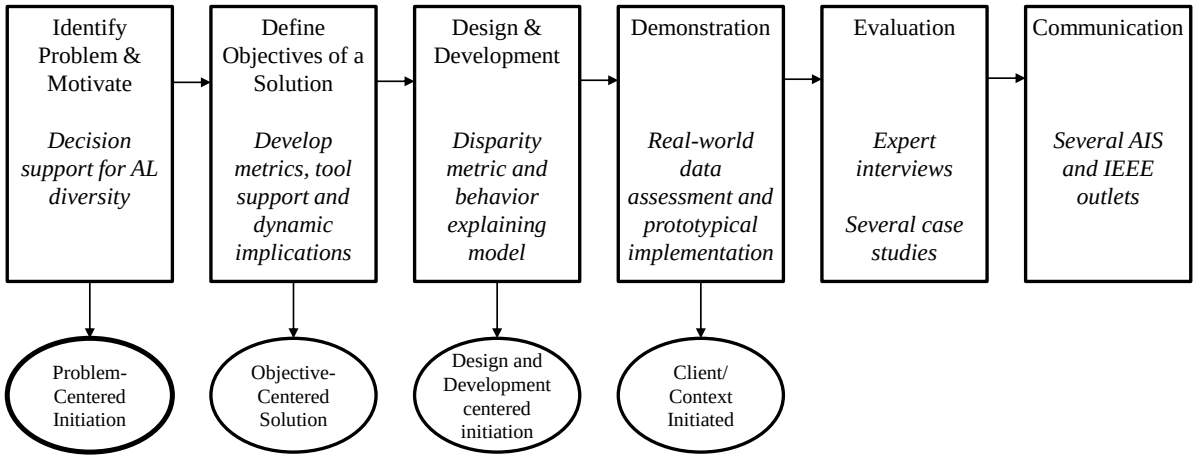


Figure 1.2.: Overall research approach based on Peffers et al. (2007)

diversity framework (see Chapter 4). This framework is used to derive the requirements for indicators quantifying diversity within the context of application landscapes. Furthermore, by considering the application landscape as a complex adaptive system additional requirements are derived which ask also for a behavior explaining model (see Chapter 6).

Subsequently, the actual design of concrete diversity indicators is performed based on available indicators within the EAM field and by adopting appropriate indicators from other fields like ecology (see Chapter 5.4.4). To enable enterprise architects to calculate these indicators for organization-specific application landscapes they are responsible for, a tool capable to do these calculations is developed. Thereby, the architecture of an existing EAM tool, i.e., Hybrid-Wikis (Matthes et al., 2011), is extended to allow performant calculations. In parallel, a system dynamics model is developed based on dynamic hypotheses derived from literature.

The demonstration of the developed indicator and tool support artifacts is accomplished by applying both to four real-world application landscape descriptions. Thereby not only their applicability is demonstrated but also their superior features compared to previously used solutions are shown. By conducting interviews with respective enterprise architects cases in which the novel indicators seem to be useful are identified.

To evaluate the artifacts described in this thesis several expert interviews are conducted. By this means, the novel indicator approach to quantify diversity as well as the developed system dynamics model are evaluated. In addition, the application of the indicator artifacts in four case studies is observed and analyzed. These case studies take place in companies from different industry sectors and cover mid-sized as well as large and globally operating companies (see Chapter 7).

Finally, this thesis communicates the research results to scientists and practitioners. In addition, several intermediate results have already been published in academic journals or presented at academic as well as practitioners conferences. The main publications relevant for this thesis include:

- The **Pattern-based Design Research** method which is used to identify complexity and diversity metric patterns in practice has been presented at the 8^{th} Design Science Research in Information Systems and Technologies Conference (DESRIST) in 2013
- An initial version of the idea how the specific enterprise environment **impacts application landscape design** also with respect to diversity has been presented at the 5^{th} Complex Systems Design and Management Conference (CSD&M) in 2014
- The **complexity framework** has been presented at the 20^{th} Americas Conference on Information Systems (AMCIS) in 2014
- **Drivers and consequences of application landscape complexity** have been published in the Zeitschrift für Conrolling Vol. 26 No. 12 in 2014
- An **overview about complexity and diversity metrics** has been presented at the 48^{th} Hawaii International Conference on System Science (HICSS) in 2015
- The **system dynamics model** has been presented at the 12^{th} International Conference on Wirtschaftsinformatik (WI) in 2015
- The **disparity indicator** has been presented at a dedicated workshop at the 8^{th} Enterprise Architecture Management Congress (EAMKON) in 2015
- The **adolescence indicator** has been submitted to the 9^{th} Multi-Conference Wirtschaftsinformatik (MKWI) in 2016

By following the research process proposed by Peffers et al. (2007) the more general guidelines for conducting design science research proposed by Hevner et al. (2004) are also respected. To provide a more detailed overview about used research methods in this thesis Table 1.1 lists the used research methods for each contribution and links them to the previously stated research questions.

1.4. Outline of this thesis

This thesis is structured in eight chapters. The main part of the thesis is organized in three parts, including **diversity perceptions**, **diversity quantification** and **diversity dynamics**. These parts are illustrated in Figure 1.3 and organize the single chapters of this thesis. Thereby, the first chapter provides an introduction to the topic, motivates the problem addressed in this thesis from both a practical and a theoretical point of view, explains the research approach, lists all contributions to the knowledge base and outlines the structure of this thesis.

In Chapter 2 (**foundations and related work**), the theoretical foundation for the research presented in this thesis is established. This includes a presentation of the body of knowledge regarding complex adaptive systems, the resource-based view of the firm and its application to IT resources and Enterprise Architecture Management (EAM). Furthermore, related work regarding IT standardization and complexity and diversity of enterprise architectures is identified by conducting a structured literature review according to the guidelines of Webster and Watson (2002). To categorize the results, a conceptual complexity framework is first developed and then

Contribution	Applied research methods	Research question
C1: Conceptual differentiation of existing notions of complexity	Argumentative-deductive reasoning (see Wilde and Hess, 2007)	RQ 1
C2: Overview about different consequences of application landscape diversity	Quantitative cross sector analysis (see Wilde and Hess, 2007) Focus group interviews (see Calder, 1977)	RQ 2
C3: Conceptual framework for application landscape diversity	Literature analysis (see Webster and Watson, 2002; vom Brocke et al., 2009)	RQ 1
C4: Indicators for application landscape diversity quantification	Goal-Question-Metric (see Basili and Weiss, 1984) Qualitative cross sector analysis (see Wilde and Hess, 2007) Expert interviews (see Bogner et al., 2009)	RQ 3
C5: Prototypical implementation of application landscape diversity indicators	Design science research (see Hevner et al., 2004; Peffers et al., 2007)	RQ 4
C6: Overview of mechanisms to influence application landscape diversity	Quantitative cross sector analysis (see Wilde and Hess, 2007) Design science research (see Hevner et al., 2004; Peffers et al., 2007)	RQ 5

Table 1.1.: Contributions and applied research methods

applied. Thereby, currently underrepresented notions of complexity are identified within the EAM domain to substantiate the motivation of this thesis. In addition, it provides a common language for research activities described in upcoming chapters.

In Chapter 3 (**diversity perceptions**), the link between two prevalent concepts—complexity and diversity—is established. Therefore, the drivers and consequences of application landscape complexity are identified by interviewing experts from industry. Then, in addition to the theoretical considerations which already consider diversity as an essential property of complexity (Page, 2011), industry experts are surveyed regarding the drivers and consequences of application landscape diversity which allows their comparison with drivers and consequences of application landscape complexity. Furthermore, goals are identified, which are currently the reason why companies increase or decrease application landscape diversity. By assessing the relevance of these goals the relevance of this thesis is again substantiated.

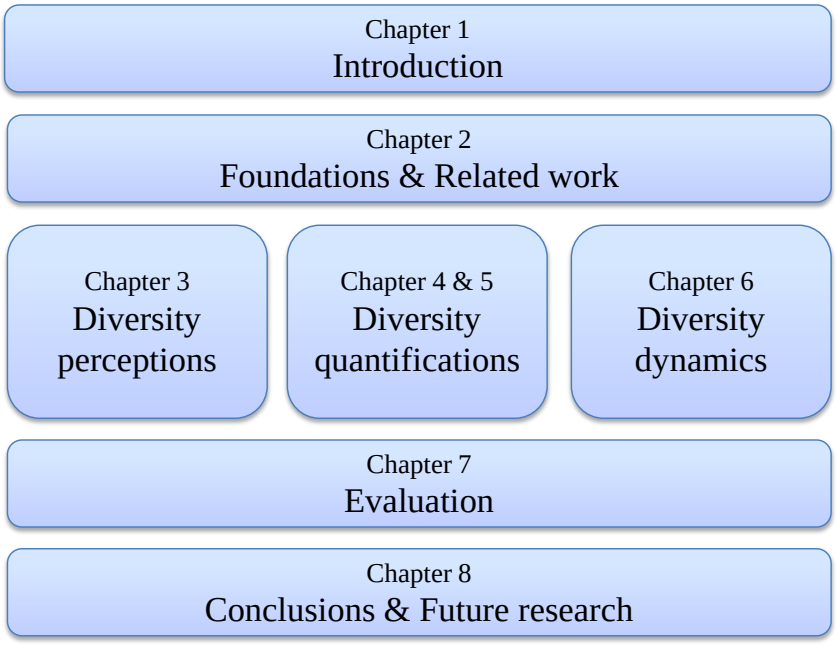


Figure 1.3.: Outline of the thesis

In Chapter 4 (**diversity framework**), a detailed conceptual framework describing different aspects contributing to diversity is developed based on existing literature. To substantiate the diversity understanding within the context of application landscapes the general framework is mapped to the concepts used to describe application landscapes. Therefore, a clear understanding of EA elements and classes of EA elements is established based on common set theory and Description Logics (DLs). By using different real-world examples, concrete instantiations of the diversity framework are described.

In Chapter 5 (**diversity quantification**), appropriate metrics are selected to quantify the different aspects belonging to the diversity framework. For these, benefits and drawbacks are assessed via expert interviews. To quantify the disparity aspect—an aspect currently neglected by EAM literature—a metric commonly used by ecologists is used and adapted to be applicable also for the context of application landscapes. In addition, a software solution is presented which implements the identified metrics and can be integrated with an existing EAM tool. To demonstrate the usefulness of both metrics and software support application landscape descriptions of four different companies are used.

In Chapter 6 (**diversity dynamics**), the prevalent notion of structural complexity is complemented by a dynamical view (see Sterman, 2000). Therefore, typically existing feedback loops within the application landscape system which influence the degree of IT standardization are identified from literature. Based on these dynamic hypotheses a system dynamics model is developed visualizing their interactions. By several expert interviews the validity of the model is assessed. The model helps enterprise architects to holistically understand the implications of their diversity decisions and can also serve as means for communication with different stakeholders.

In Chapter 7 (**evaluation**), the previously developed artifacts, i.e., diversity indicators, tool support and system dynamics model, are evaluated to assess their applicability and utility. The evaluation takes place in cooperation with four different companies. These companies belong to different industry sectors, e.g., financial services and production, and vary in size. In each company a case study is conducted in which the artifacts are adapted to organization-specific needs. The successful application of the developed artifacts demonstrates their usefulness.

In Chapter 8 (**conclusion and outlook**), the contributions of this thesis are summarized and critically reflected. Thereby, possible limitations are identified and open questions for future research are proposed.

CHAPTER 2

Foundations and related work

"It's all about fundamentals. You've got to get fundamentals down because otherwise the fancy stuff isn't going to work."

Randy Pausch, *The Last Lecture*

To lay down the foundation of this thesis the research results of several related disciplines are outlined in this chapter. First, an introduction to system theory is given focusing on complex systems and different notions of complexity found in literature. Then, Enterprise Architecture Management (EAM)—a discipline trying to improve Business-IT alignment—is outlined in particular with respect to related work on IT standardization. Due to the fact that IT assets are viewed as valuable resources in this thesis the resource-based view of the firm is also introduced. Special emphasis is directed to the notion of digital options created by IT resources. Finally, related work on complexity and diversity of enterprise architectures is summarized and categorized using a self-created framework.

2.1. Complex adaptive systems

Complexity—an emergent property of a system—can be found in various different research areas (Mainzer, 2007). To lay the foundation for this thesis a short overview about general systems theory (see Bertalanffy, 1968) as well as complex systems theory (see Weaver, 1948) is provided. Furthermore, the fundamentals of socio-technical systems are reviewed because an application landscape (as defined in Chapter 1.1) falls into this category of systems.

2.1.1. General systems theory

The idea of a theory for systems has first been formulated by Bertalanffy (1968). He states that "there exist models, principles, and laws that apply to generalized systems or their subclasses, irrespective of their particular kind, the nature of their component elements, and the relationships or "forces" between them. It seems legitimate to ask for a theory, not of systems of a more or less special kind, but of universal principles applying to systems in general" (Bertalanffy, 1968, p. 32). But to apply such theory of systems we need to know what a system is. Unfortunately, there is no consistent definition established among the different research communities. Nevertheless, systems are used to describe problems in various disciplines, e.g. social sciences (see Luhmann, 1984) and politics (see Waltz, 2010). For this thesis, the following definition of a system will be used which is in line with the perceptions of Wiener (1965); Bertalanffy (1968); Bunge (1979); Luhmann (1984) and Bossel (2004a).

Definition: System
A system is a set of **elements** and their **interrelations** which are delimited by their **environment**. Such elements can be tangible or intangible components forming the system by their interactions which address a common goal. A system is considered to be open if interaction with the environment can occur.

According to this definition, Figure 2.1 depicts the different aspects of a system. In the center, the different elements of a system are indicated by boxes. These elements interact with each other via feedback. Together, system elements and their feedback patterns form the structure of a system. The system boundary indicated by a bold line separates the system from its environment and therefore from other systems. As indicated by arrows, the environment can have various different impacts on a system. Likewise, the system itself can impact its environment.

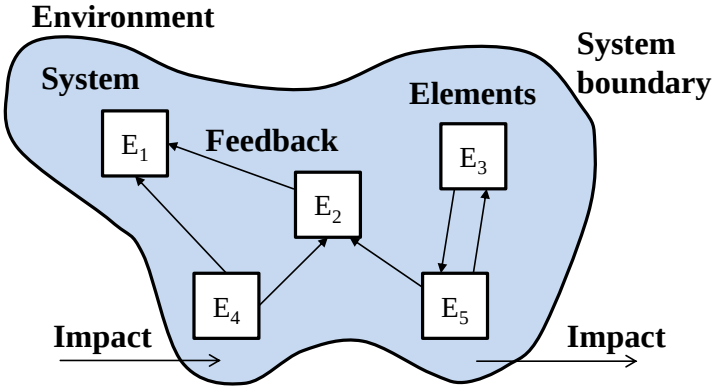


Figure 2.1.: Constituents of systems based on Bossel (2004a, p. 36)

According to Luhmann (1984, p. 15), general system theory and their interdisciplinary efforts might form a paradigm shift in sense of Kuhn (1962). Based on the foundation of general systems theory many other research streams evolved. These include, for example, cybernetics (see Ashby, 1956; Beer, 1967; Wiener, 1965) and complex systems theory (see Weaver, 1948) whereas the

latter laid the foundation for topics like self-organization, emergence and autopoiesis. Merali and McKelvey (2006) argue that complexity science might be able to initiate a paradigm shift also in the field of information systems research. The following section will introduce the concepts of complex systems theory relevant for this thesis.

2.1.2. Complex systems theory

In the glamor of various success stories complexity science is still an interdisciplinary research area without a commonly agreed definition of complexity (Johnson, 2007). Therefore, it is not surprising that different views of complexity emerged over time. To get a concise overview about different notions of complexity a multi-dimensional framework unifying the most prevalent views on complexity is developed in this section (see also Schneider et al. (2014)). Explicating one's notion of complexity can avoid misunderstandings and identify underrepresented notions within a specific field of research.

Organized and disorganized complexity

The idea of complexity within systems goes back to Weaver (1948). He was the first distinguishing between simple systems, systems of organized complexity and systems of disorganized complexity. Thereby, problems or systems of simplicity are mostly concerned with few variables which applies, for example, to machines like radios or automobiles. Thus, respective questions about such systems can be answered with standard maths. By contrasts, problems of disorganized complexity focus on a very large amount of variables which qualify for the application of probability theory and of statistical mechanics. The following two examples given by Weaver (1948) explain the difference.

Example 2.1: Simple systems. Imagine a single ivory ball on a billiard table. A simple problem is analyzing and predicting the motion of the ball as it moves about on the table.

Predicting the movement in such scenario is quite simple. Not many variables need to be taken into account and standard maths can be applied. However, this is different in next example.

Example 2.2: Disorganized complex system. Consider a large billiard table with millions of balls rolling over its surface colliding with one another and with the side rails.

In order to predict the movement of any single ball, a large amount of variables needs to be tracked. Although the detailed history of each ball's movement might not be traceable, certain important questions can be answered by using probability theory and of statistical mechanics:

- On the average how many balls per second hit a given stretch of rail?
- On the average how far does a ball move before it is hit by some other ball?
- On the average how many hits per second does a ball experience?

The third kind of systems distinguished by Weaver (1948) are systems of organized complexity. Regarding the number of variables, these systems form a middle region between the other two. In addition, problems within such systems, as contrasted with the disorganized situations with which statistics can cope, show the essential feature of organization. Such questions include, for example:

- Why does salt water fail to satisfy thirst?
- Why does the amount of manganese in the diet affect the maternal instinct of an animal?
- Why is one chemical substance a poison when another, whose molecules have just the same atoms but assembled into a mirror-image pattern, is completely harmless?

For problems like these, statistical methods do not hold the key. They involve dealing simultaneously with a sizable number of factors which are interrelated into an organic whole. The next example illustrates these aspects.

Example 2.3: Organized complex system.
On what does the price of wheat depend?

To answer this question, a substantial number of relevant variables needs to be taken into account and they are all interrelated in a complicated, but nevertheless not in an arbitrary fashion.

Qualitative and quantitative complexity

Qualitative complexity refers to the qualitative evaluation of a certain attribute or variables of a system. In such case, the system or problem under consideration is considered to be complex implying a set of possible other characteristics or to be not complex, i.e. simple. A good example for a qualitative notion of complexity is the "El Farol Problem" described by Arthur (1994).

Example 2.4: Qualitative complexity and the El Farol Problem. In this multiple-stage game, participants have to decide whether or not to visit

> a bar in each round. All "players" prefer to enjoy a drink at the bar rather than staying at home, but the bar has a maximum capacity of seats. For the players, it is less enjoyable to attend an overcrowded bar than staying at home. In each round, each player has to decide to attend the bar or not. After the round it can be determined whether he/she is better off doing so.
>
> ⌟

In this artificial game—which is actually very close to many real world situations—the success of each participant's decision depends on the decisions of all other players. Remembering decisions of others will not provide new insights for the next round since the participants' decisions might change in each round. Therefore, this problem is considered to be a complex problem. The type of complexity is independent from the number of players, the number of rounds or the memory capacity of players. A qualitative notion of complexity is also used by researchers studying complex system phenomena such as self-organization (see Kauffman, 1993), emergence (see Anderson et al., 2002) or dynamical systems (see Gardner, 1970). Thereby, the authors attribute complexity to the system under consideration without trying to calculate how complex a system is.

The qualitative notion of complexity is also prevalent among researchers from the fields of cybernetics. Achievements there include, for example, Ashby's law or requisite variety. In this law it is stated that only "variety can destroy variety" (Ashby, 1956, p. 207). This leads to the insight that every good regulator of a system must have a model of that system (Conant and Ashby, 1970) and have at least equal variety. Likewise, Beer (1967) founded the field of management cybernetics by building the Viable System Model (VSM) based on Ashby's law. Thereby, a viable system is organized in a way as to meet the demands of surviving in a changing environment. As visualized in Figure 2.2, it consist of five interacting subsystems, operation, coordination, control, planning and identity, and has been applied in various contexts, e.g. project management (see Britton and Parker, 1993), organizational modeling (see Brocklesby and Cummings, 1996) and enterprise architecture management (see Buckl et al., 2009).

- **System one** (operation) contains the primary activities of the system under consideration, which directly interact with the environment. Entities need to be viable by them self.
- **System two** (coordination) includes the information channels and bodies, which ensure that the primary activities of system one work in coordination. Here, self-organization occurs.
- **System three** (control) represents the structures and controls, which establish the responsibilities and rights to maintain the resource allocation of system one.
- **System four** (planning) is concerned with a holistic and future-oriented perspective to support strategic decision making.
- **System five** (identity) is responsible for managing the overall policy decisions. It should provide clarity about the overall direction, values, and purpose of the system under consideration and serves as a mediator between systems three and four.

In addition to the qualitative notion of complexity, other researchers apply a quantitative notion

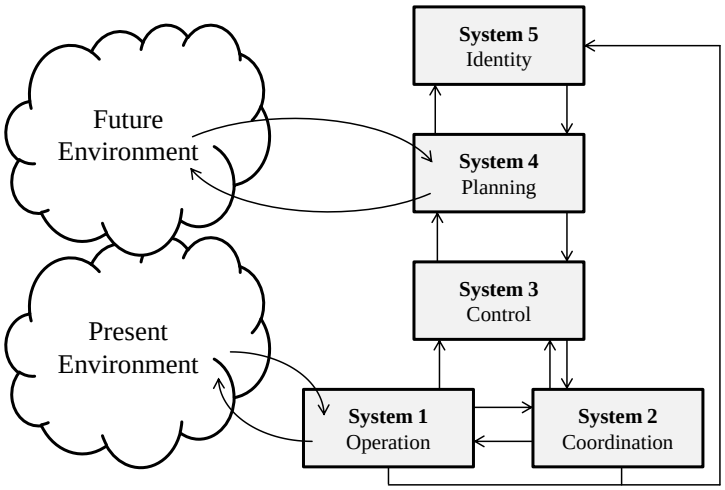


Figure 2.2.: The viable system model according to Beer (1967)

of complexity. For example, a classic measure of complexity has been proposed by Kolmogorov (1963). It describes the length of the shortest computer program capable of generating a given string. Another well-known quantitative measure is entropy (Shannon, 1948) which can be understood as a measure for uncertainty in a message. Other approaches have been developed to measure (computing) complexity as well, for instance, based on the number and variety of both components and their interactions within a system (Schneberger and McLean, 2003). In computer science the complexity of an algorithm is usually determined as a function of its input length. Algorithms are classified according to their asymptotic behavior for large inputs using Landau notation (Bachmann, 1894). To determine algorithmic complexity, the number of calculations or the amount of memory consumption is typically of interest.

Subjective and objective complexity

Another way to distinguish different notions of complexity is to demarcate the subjective from the objective notion. Objective complexity refers to a notion of complexity that is independent from the observer of a system. Complexity is considered to be a property of the system under observation, much in the same way as mass or volume of a physical body (Fioretti, 1999). Such objective views are prevalent, for example, in the domain of qualitative complexity where system properties like emergence (see Anderson et al., 2002) are investigated. The same applies to many complexity metrics as their results are free of individual influence (Landauer, 1988).

However, complexity can also be considered to be a property of the relationship between a system and its observer (Rosen, 1977). If an observer perceives a system as complex, his/her mental model of the system cannot explain his/her observations. In contrast to the objective notion of complexity, the subjective notion requires an individual observing the system. Researchers define subjective measures, for example, based on mental categories of the observer (see Fioretti, 1999) or as being composed of other objective measures (Flückiger and Rauterberg, 1995, csee[).

Structural and dynamic complexity

Another important notion of complexity is structural complexity, which is also known as combinatorial or detail complexity (Sterman, 2000). It covers a pattern of system components, i.e. the number of variables as well as the cause-and-effect-relationships between them. A structural notion of complexity is employed, for example, in network research where cyclic groups, spanning sub-graphs and extended connectivity play an important role (see Bonchev and Buck, 2005). The two well-known measures of complexity, i.e. Kolmogorov complexity and Shannon entropy, also apply a structural notion of complexity.

In contrast, dynamic complexity refers to the observation of multifaceted interdependencies as well as changes of interactions between variables of a system. According to Sterman (2000, p. 21), "dynamic complexity arises from the interactions among the agents over time". In complex (dynamic) systems, the impact of actions often cannot be reversed; therefore a comparison of system states of the past and the present is rather difficult. With several interacting feedbacks, determining an exclusive effect of a certain variable is hardly possible because it is likely that other variables change as well. As a consequence, system behavior interpretation is usually complicated. Additionally, delays in cause and effect have to be considered which can result in system instability and influence the dynamics of a system. Dynamic complexity arises, for example, when systems are heavily interacting with each other and the natural world or if actions influence future choice options (Sterman, 2000). The dynamic complexity notion has also been applied in the socio-economic discipline (Forrester, 1961).

2.2. Enterprise architecture management

When referring to Enterprise Architecture Management (EAM), it is necessary to distinguish between the object under consideration, i.e. an enterprise architecture, and a respective management approach called enterprise architecture management or enterprise architecting. To form an unambiguous foundations of used terms, both will be outlined in the next sections. In addition, the main goal of EAM, i.e. achieving Business-IT alignment, will be briefly explained. Finally, related work on the topic of IT standardization and diversification is described as it forms the basis of application landscape diversity management.

2.2.1. Enterprise architecture

The idea of documenting the an enterprise architecture dates back to the 1980s when Zachman (1987) published the first EA framework. Thereby, the goal of an enterprise architecture is—as indicated by its name—to describe the architecture (system structure) of an entire enterprise. The major differentiator is that it tries to integrated previously existing modeling approaches used in different organizational units separately. Thereby, it addresses, for example, business as well as IT related aspects. Up to now, literature provides several different definitions for enterprise architectures. Extensive surveys regarding different definitions of EA have been conducted, for example, by Schönherr (2008) and Kotusev et al. (2015). For example, a popular definition has been developed by the MIT Center for Information System Research.

> The enterprise architecture is the organizing logic for business processes and IT infrastructure, reflecting the integration and standardization requirements of the company's operation model (Ross et al., 2006, p. 9).

Many other enterprise architecture definitions (see Winter and Fischer, 2007; Buckl et al., 2007) are build upon the ISO/IEC/IEEE 42010 standard defining architectural descriptions of software-intensive systems.

> [An enterprise architecture is 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 (International Organization for Standardization, 2007).

Accordingly, the definition of an enterprise architecture used throughout this thesis is built upon the ISO standard and related work (International Organization for Standardization, 2007; Buckl, 2011).

Definition: Enterprise Architecture (EA)
EA is 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.

It is important to mention, that "every system and organization has an architecture" (Rechtin, 1999, p. 175) no matter whether it is documented or not. If such architecture should be documented, a layered approach proved to be useful (Winter and Fischer, 2007; The Open Group, 2011). Because literature not yet agreed upon the amount of layers used to describe an EA or their specific names, one specific approach suitable for this thesis is chosen. As visualized in Figure 2.3, the approach presented by Buckl (2011) consists of three layers. These include the **business & organization** layer, the **application & information** layer and the **infrastructure & data** layer. Displayed vertically, it also contains cross-cutting aspects including **visions & goals**, **strategies & projects**, and **principles & standards**, representing elements which are not part of, but exert influence on any of the elements organized in layers. In addition, abstraction layers are used to encapsulate functionalities of the underlying layer or cross-cutting aspect. Besides the different elements grouped into layers and cross-cutting aspects, the relationships between these elements are essential because they reflect the alignment between business and IT.

Based on this conceptual understanding of an EA and complex systems (see Chapter 2.1.2) the preliminary definition of an application landscape provided in Chapter 1.1 can be further elaborated. In addition, an overview of typical elements of an application landscape can be found in Chapter 5.1.

Definition: Application landscape
The application landscape covers all components of the application and infrastructure layer as well as their relationships. Furthermore, people directly interacting with these components by using or designing them also belong to the extended application landscape system.

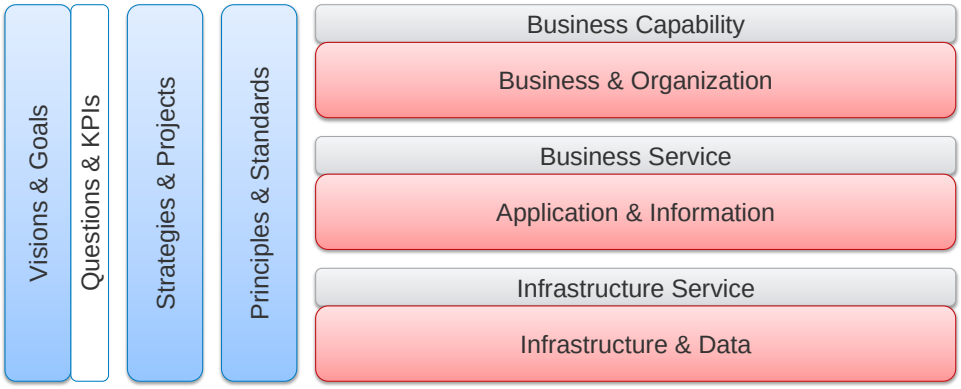


Figure 2.3.: Layered approach to EA documentation according to Buckl (2011)

Embodied in the enterprise architecture the application landscape forms an open system which is influenced by the rest of the organization, i.e. its environment, and simultaneously impacts the organization. The goal of the function or process managing an Enterprise Architecture (EA) is, among others, to align these mutual impacts by considering the systems behavior and changing its structure if required. The next section provides an overview about other goals of EAM as well as typical approaches.

2.2.2. Enterprise architecture management and business-IT alignment

Within Enterprise Architecture Management (EAM) research, a great diversity regarding goals, concepts and used languages exists. For example, Schelp and Winter (2009) analyze different language communities within the context of EAM research. Their analysis framework distinguishes between a meta-model which is used to describe an EA and a procedure model specifying how an EA is managed. The research framework is used to analyze the contributions of seven major academic groups. As a result, the authors conclude that most of the research groups constitute local language communities with a self-developed language and method. The same diversity can also be found in practice where organizations pursue different goals with their EAM function (see Buckl et al., 2010) and apply different methods and visualizations (Buckl et al., 2008; Matthes et al., 2015). In addition, this diversity is reflected by the vast amount of EAM frameworks developed by researchers (see Frank, 2002; Wegmann, 2002; Aier et al., 2009; Buckl et al., 2010), practitioners (see Lankhorst, 2009; Niemann, 2005; Schekkerman, 2008; Keller, 2012; Hanschke, 2013), public institutions (see MoD, 2008; DoD, 2009) and standardization bodies (see The Open Group, 2011). Given the fact that no agreed-upon definition of EAM has been developed until now the definition of Buckl (2011) will be used because it is in line with the understanding of an enterprise as complex system (Buckl et al., 2009).

Definition: Enterprise Architecture Management
Enterprise Architecture Management (EAM) is a continuous and self maintaining management function seeking to improve the alignment of business and IT and to guide the managed evolution of an (virtual) enterprise. Based on a holistic perspective on the enterprise furnished with information from other enterprise-level management functions it provides input to, exerts control over, and defines guidelines for these enterprise-level management functions (Buckl, 2011).

According to this definition, EAM functions have to regard the enterprise as a whole and form a continuous approach targeting at an improvement of the alignment of business and IT. As visualized in Figure 2.4, EAM functions need to exchange data and interact with other enterprise management disciplines like demand, portfolio and strategy management while acting like a "glue" (Buckl et al., 2009). In addition, the definition given above emphasizes that the EAM function should guide the managed evolution as defined by Murer et al. (2008). The idea here is not to set specific goals for each project within an enterprise. Instead, an evolution channel should be defined which ensures that each modification to the EA results in both agility and business value gains. This is also in line with viewing the enterprise as complex system in which the absence of central control is always a feature (Ladyman et al., 2013). By refraining from putting an EAM function in place to act like a central control for the enterprise architecture other management disciplines like demand and project management keep their influence on the actual design.

If an organization decides to implement an EAM function, different values can be achieved. For example, Niemi (2008) analyzed existing literature on EAM benefits and validated identified benefits with focus group interviews with practitioners. Thereby, the author identified at least 27 different benefits an organization can get from executing EAM. Figure 2.5 categorizes these benefits according to being measurable and attributable to EAM.

A complementary study on the question how EAM leads to organizational benefits has been conducted by Tamm et al. (2011). Therein, four different benefit enablers are identified: organizational alignment, information availability, resource portfolio optimization and resource complementarity. For example, the organizational alignment enabler covers benefits like integrated view of the enterprise and improved communication. The information availability enabler covers aspects like shared reference information and improved understanding of resources and processes. The resource portfolio optimization enabler covers aspects like discovery and elimination of redundancy. The resource complementarity enabler consists of aspects like improved resource integration and enhanced interoperability. The respective theory developed by Tamm et al. (2011) distinguishes between benefits that can flow from EAM directly and benefits that can be achieved only through the implementation of the EA plans, i.e., from an EA-guided operating platform. Furthermore, the extend to which these four benefit enablers can be realized depends on the EA quality which can be assessed, for example, by the following aspects: Appropriate scope-detail-cost balance, appropriate CIO skills, clear and agreed-upon architecture principles, documentation tools, effective project management, skilled architecture team, sufficient (on-going) funding and stakeholder acceptance & involvement. Given this general overview about EAM and its potential benefits, in the following, the main goal of EAM, i.e. fostering business-IT alignment, is explained in more detail.

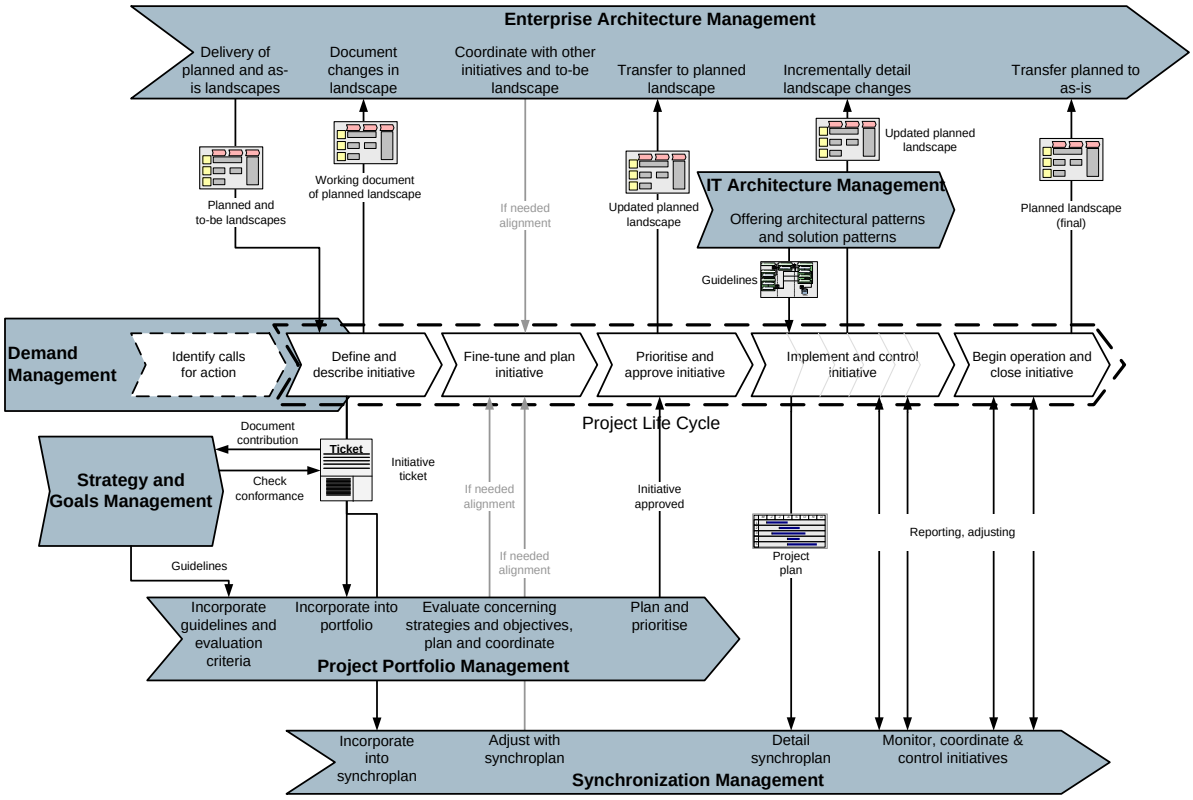


Figure 2.4.: The EAM function acting as a glue according to Buckl et al. (2009)

According to Hevner et al. (2004, p. 78) business-IT alignment is "central to the IS discipline". Various studies showed that business-IT alignment affects profit, productivity, sales growth and reputation (Chan et al., 1997, 2006; Oh and Pinsonneault, 2007; Tallon, 2007; Preston and Karahanna, 2009). Furthermore, "sound business-IT alignment is a key prerequisite for business processes to be supported effectively by information technology, especially in large organizations" (Murer et al., 2008, p. 163). This includes, for example, that business requirements towards IT are defined in a systematic way, that IT departments understand and accept these requirements and that they deliver IT solutions fulfilling these requirements. Enterprise Architecture Management (EAM) becomes essential to ensure business-IT alignment because the effort for operating and maintaining a growing application landscape may grow even faster than linear with the number of applications (Murer et al., 2008, p. 163). Therefore, the following tasks need to be carried out: requirements need to be prioritized system-wide according to the business and IT strategy, projects need to be defined with clear not-overlapping scopes and ensuring that each project contributes to the target architecture envisioned by the architects and stakeholders. It is important to note that "alignment evolves into a relationship where the function of IT and other business functions adapt their strategies together" (Luftman, 2000). Furthermore, achieving business-IT alignment requires, for example, support from senior management, strong leadership, appropriate prioritization and thorough understanding of both business and technical environments. The holistic and structured approach of EAM is one means to achieve business-IT alignment (Winter and Fischer, 2007; Wegmann, 2002; Zarvic and Wieringa, 2006).

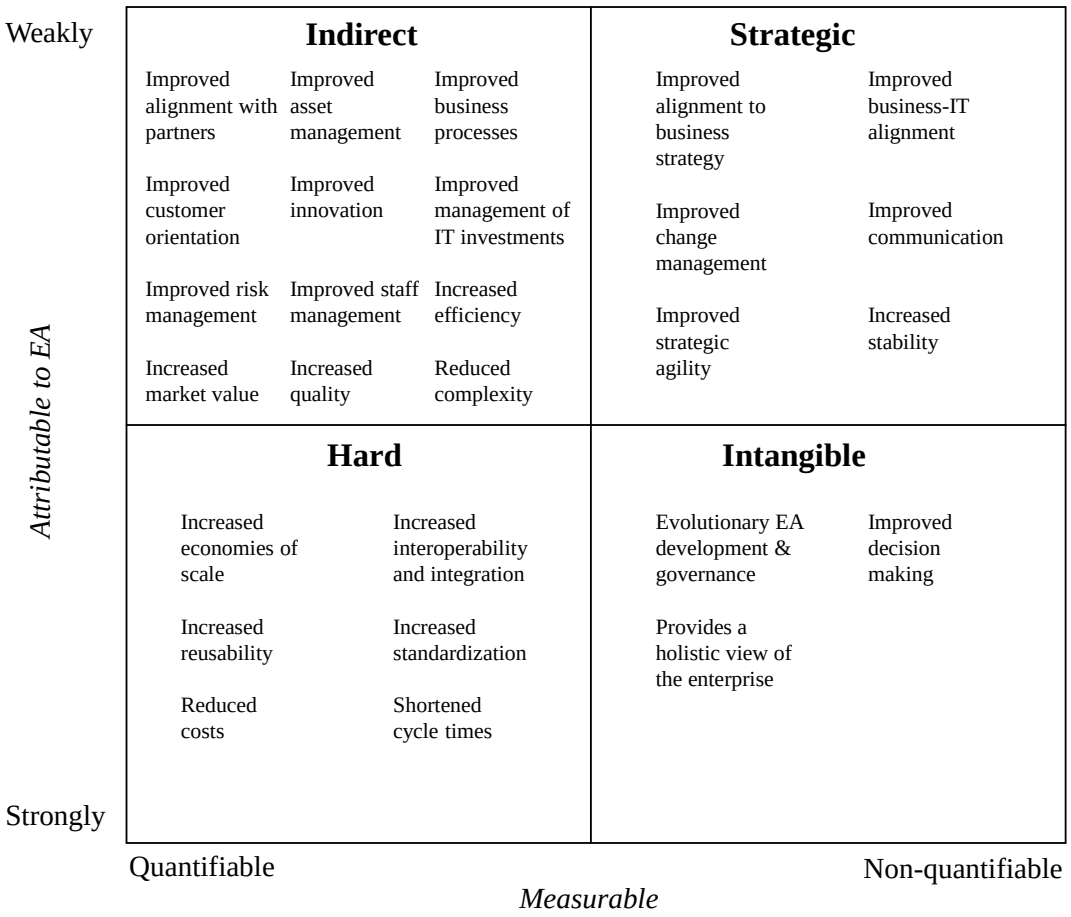


Figure 2.5.: Categorized EAM benefits according to Niemi (2008)

2.2.3. IT standardization and diversification

IT standardization is a prevalent means in Enterprise Architecture Management (EAM). After decades of uncontrolled growth a standardized IT landscape forms the first level of maturity in EAM endeavors (Ross et al., 2006). Especially literature from practitioners presents IT standardization as a core task of enterprise architects (Niemann, 2005; Keller, 2012; Hanschke, 2013). The use of company-wide standards to manage IT resources is also advocated by prominent standardization bodies (The Open Group, 2011). The goal of EA standards is to guide IT departments and business units in their technical choices and project-level decisions related to data and application design (Boh and Yellin, 2007). Cost reduction through economies of scale (DeSanctis and Jackson, 1994) is the most prominent and frequently mentioned expectation of IT standardization activities. Another important reason for standardization is interoperability which is especially required when new business models should be implemented requiring integration among business units (Tanriverdi, 2005). However, little empirical research has been conducted to verify these claims. On the contrary, IT standardization can also result in drawbacks, e.g., it restricts business units resulting in less optimal IT solutions (Ross et al., 2006; Boh and Yellin, 2007) and hinders exploratory activities. In addition, organizations need

to commit substantial amounts of time and resources to create, implement, and maintain EA standards (Kim and Everest, 1994). To cope with a changing environment, organizations need redundant processes and local autonomy, i.e. regional deviations from standards have to be tolerated (Bossel, 2004a; Schneider and Matthes, 2014a). Hence, enterprise architects face the challenge of defining an appropriate degree of diversity within the IT landscape to provide appropriate solutions for business departments while keeping costs for running the IT in mind (see Murer et al., 2008). In general, the ability to pursue disparate things at the same time is called ambidexterity (Gibson and Birkinshaw, 2004) which in this context requires to achieve both IT standardization and IT differentiation (Gregory et al., 2015). This will be referred to as application landscape landscape diversity management. According to the enterprise ecological adaptation school of thought (Lapalme and de Guerre, 2013), EA must contribute to fostering innovation and enterprise-in-environment co-evolution; hence managing diversity is an important concern that goes beyond cost reduction. It is important to note that EA standards can be set to manage different kinds of IT resources. For example, Boh and Yellin (2007) distinguish the following types:

1. EA standards for physical IT infrastructure management
2. EA standards for human IT infrastructure management
3. EA standards for integrating business applications
4. EA standards for enterprise data integration

The first category considers IT infrastructure elements, i.e underlying technologies required to run business applications such as operating systems, database management systems or web servers. The second category regards standards to manage human IT resources, e.g., organizational IT skills, expertise, competencies and knowledge. The third category targets at the standardization of the application portfolio and the fourth category on the definition of standards for data elements. The focus of this thesis is on technical elements of the enterprise architecture. This includes the first as well as the third category of standards mentioned by Boh and Yellin (2007).

Up to now, the consequences of standardizing IT are not fully understood. For example, there is are no longitudinal comparative studies available in which the impact of standardization on costs is measured. Furthermore, EAM literature uses inconsistent terms when considering the diversity within application landscapes and lacks a unified conceptual understanding of diversity (see Chapter 5.2). Apart from that, technology diversification has been identified as a driver for business growth (Granstrand and Oskarsson, 1994). Nevertheless, it remains unclear in how far this holds also true for the diversification of IS or IT components.

2.3. Resource-based view of the firm

A major research question for the economics discipline and especially the strategic management discipline is how companies can create competitive advantage. A popular and often applied theory—the resource-based view of the firm (Wernerfelt, 1984)—tries to explain competitive advantage with superior resources a company has access to. When managing IT resources with

respect to their diversity impacts on competitive advantage cannot be ruled out. Therefore, this chapter describes the resource-based view as well as its implications for an enterprise's application landscape. Furthermore, the concept of organizational agility (see Tallon and Pinsonneault, 2011)—a capability allowing enterprises to react on a changing environment—is introduced because it explains why enterprises need to perform exploitative (e.g. IT standardization) as well as exploratory (e.g. IT differentiation) tasks whereas both require managing diversity.

2.3.1. Competitive advantage based on superior resources

Wernerfelt (1984) coined the term of the resource-based view of the firm—a theory relating competitive advantage to superior resources the company has access to. Elements of the theory can also be found in earlier works, e.g. in Penrose (1959). In contrast to other theories at that time, the resource-based view theory shifts the focus from the product side of a company to the resource side. The resource perspective seems to provide a basis for addressing some key issues in the formulation of a strategy for diversified companies, such as (Wernerfelt, 1984):

1. On which of the firm's current resources should diversification be based?
2. Which resources should be developed through diversification?
3. In what sequence and into what markets should diversification take place?
4. What types of firms will it be desirable for this particular firm to acquire?

Initially, the concept of "resources" covered tangible and intangible assets tied semipermanently to the firm including, for example, brand names, in-house knowledge of technology, trade contacts, machinery and capital. As the theory evolved, resources have been distinguished from capabilities (Amit and Schoemaker, 2012). Resources are tradable and non-specific to the firm and are converted to final products or services. Capabilities, in contrast, are firm-specific and are used to combine resources within the firm using organizational processes. Because not all resources and capabilities might be able to create competitive advantage, Amit and Schoemaker (2012) define *strategic assets* as the set of difficult to trade and imitate, scarce, appropriable and specialized resources and capabilities. Accordingly, Figure 2.6 shows desired characteristics of the firm's resources and capabilities.

Making decisions regarding strategic assets, i.e. resources and capabilities, occurs ordinarily in a setting that is characterized by (1) Uncertainty about (a) the economic, industry, regulatory, social, and technological environments, (b) competitors' behavior, and (c) customers' preferences; (2) Complexity concerning (a) the interrelated causes that shape the firm's environments, (b) the competitive interactions ensuing from differing perceptions about these environments; and by (3) Intraorganizational conflicts among those who make managerial decisions and those affected by them (Amit and Schoemaker, 2012). Nevertheless, strategic assets are considered to be a primary reason for competitive advantage implying that management has to make the right decisions although the mentioned conditions of uncertainty, complexity and conflict are usually hard to articulate or model. Figure 2.7 summarizes the key concepts of the resource-based view of the firm based on Barney (1991) and Amit and Schoemaker (2012). In the next chapter, the concept of organizational agility is introduced as new business imperative and applications of

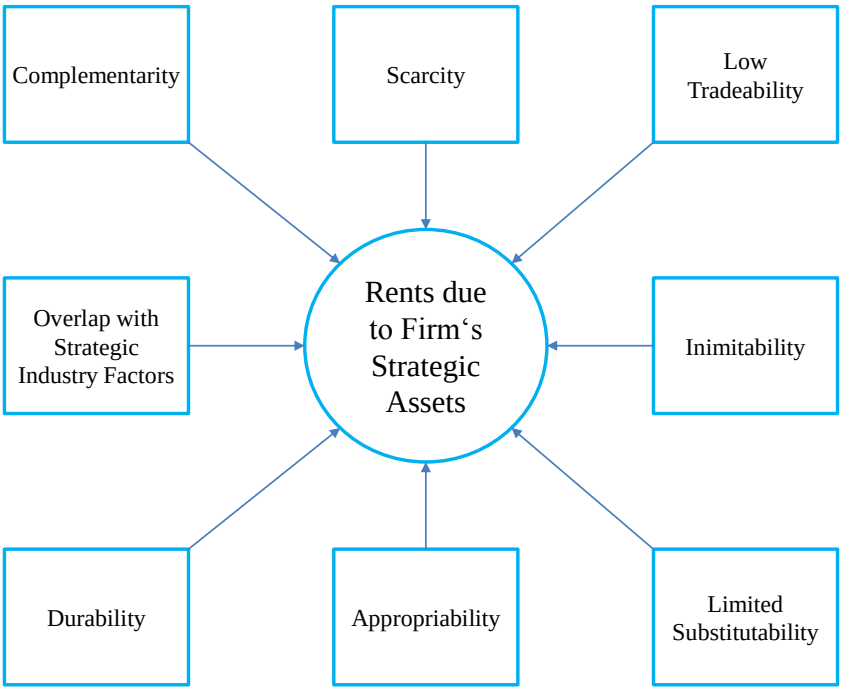


Figure 2.6.: Desired characteristics of the firm's resources and capabilities according to Amit and Schoemaker (2012)

the resource-based view in the domain of IS are described to highlight how IT resources can create organizational agility and therefore competitive advantage.

2.3.2. Organizational agility and digital options

The resource-based view of the firm theory presented in the previous chapter explains how resources and capabilities can create competitive advantage. Thereby, the impact of environmental change is underrepresented within the theory although it is this kind of change separating *good* from *bad* resource allocation decisions. The dynamic capabilities framework (Teece et al., 1997) extends the resource-based view to incorporate environmental and technological change. Teece et al. (1997, pp. 522-523) argue that "a firm's previous investments and repertoire of routines constrain its future behavior;" and that "opportunities for learning will be 'close in' to previous activities and thus will be transaction and production specific". This implies that firms invest in particular types of resources and learn how to use those resources over time by developing asset-specific skills and accompanying routines (Cohen and Levinthal, 1990). Due to the fact that today enterprises are faced with changing environments and increasing uncertainty due to rapid development of new technologies, increasing globalization of markets and the rise of innovative new forms of organization (Sanchez, 1997; Jones, 2010), *agility* has emerged as a key business imperative (Tallon and Pinsonneault, 2011). In this context, agility can be defined as follows.

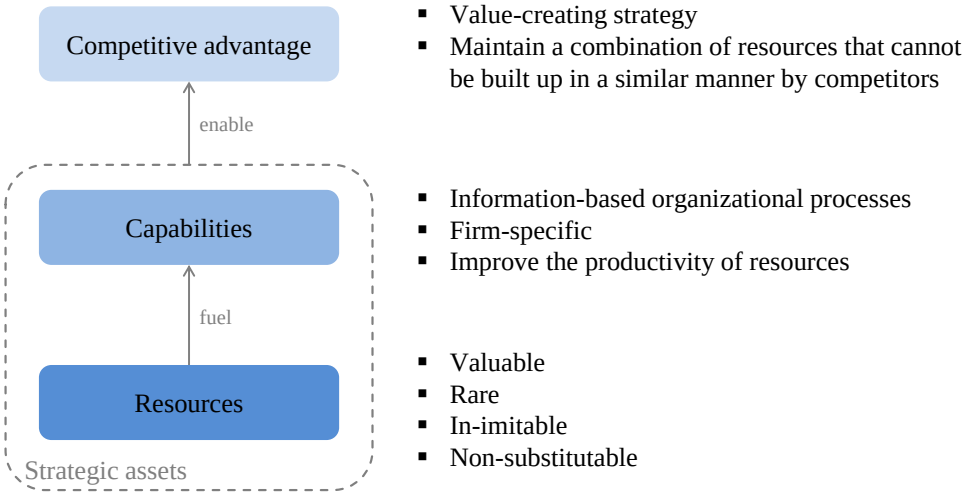


Figure 2.7.: Key concepts of the resource-based view of the firm

Definition: Organizational agility
Organizational agility is defined as the ability to detect and respond to opportunities and threats with ease, speed, and dexterity.

(Tallon and Pinsonneault, 2011)

Up to now, several IS researchers analyzed in how far the design of IT systems impacts organizational agility. For example, Galliers (2006) argues that "while the alignment concept may be intuitively appealing, an issue that has remained relatively unchallenged and unquestioned is how to align a relatively fixed ICT that is implemented in an organization with a business strategy and associated information requirements that are in constant need of adjustment, while keeping in line with the dynamic nature of the organization's business imperatives". Consequently, IT systems have to be agile. Another example is provided by Weill et al. (2002) who investigate the enabling role of IT infrastructure for organizational agility. Based on the analysis of 180 electronically based business initiatives they conclude that agility requires time, money, leadership and focus and that successful companies have a clear picture about their overall infrastructure capability, i.e. a detailed and up-to-date EA description, and how each incremental investment contributes to it.

To enhance their agility, organizations rely progressively on IT (Sambamurthy et al., 2003), whereby IT emerges as a strategic differentiator. Ferrier et al. (1999) argue that firms possessing a more complex base of resources and capabilities are in an advantageous position to launch competitive actions. Thereby, agility requires the exploration and exploitation of opportunities for market arbitrage. Exploratory activities target long-term success by conducting research and experiments whereas exploitative activities target short-term success by establishing and refining routines (Levinthal and March, 1993). In other words, exploitative innovations involve improvements in existing components and build on the existing technological trajectory, whereas exploratory innovation involves a shift to a different technological trajectory (Benner and Tushman, 2002). Hence, the requirements regarding the diversity of IT systems differ across areas

within the application landscape whether they are subject to exploration or exploitation. This is also in line with general systems research describing the tension of systems orienting towards scarce resources, i.e. efficiency or exploitation, or changes in the environment, i.e. adaptivity or exploration (Bossel, 2004a; Schneider and Matthes, 2014a). The tension between standardization, an activity related to exploitation, and flexibility which is required to adapt to a changing environment has been addressed by many IS researchers (Hanseth et al., 1996; Kettinger et al., 2010) and will be further discussed in Chapter 3.2.

To explain how IT resources can actually create competitive advantage through creating agility and therefore qualify for a comprehensive diversity management approach, Sambamurthy et al. (2003) draw on the concept of *digital options* which are the result of IT infrastructure investments. In this context, digital options can be considered as rights to future investment choices based on an initial expenditure without a current obligation for full investment.

Definition: Digital options
Digital options are a set of IT-enabled capabilities in the form of digitized enterprise work processes and knowledge systems.

(Sambamurthy et al., 2003)

According to Amram and Kulatilaka (1999), when path dependencies in the form of prior learning, investment or experience guide prospects for exploiting emergent opportunities (as outlined before), the holding of options is economically advantageous. In addition, Fichman (2004) argues that "firms that make such investments retain full exposure to the upside potential of the technology should subsequent events prove favorable to deployment, but they can limit losses to just the positioning investment if future events prove unfavorable". Accordingly, increasing the diversity among IT systems can be regarded valuable as long as new digital options are created and technology choices are made based on entrepreneurial alertness. As visualized in Figure 2.8, the theory presented by Sambamurthy et al. (2003) distinguishes between two fundamental processes. First, the capability building processes integrate IT and business resources into organizational capabilities. Thereby, entrepreneurial alertness facilitates the conversion of IT investments and capabilities into digital options because strategic foresight is vital so that executives can anticipate the opportunities and business value available in information technologies. Second, entrepreneurial action processes make use of provided digital options to foster creative combination of agility and entrepreneurial alertness for the launch of competitive actions. For example, companies having all three kinds of agility enabled through digital options will detect a promising opportunity for product innovation through customer agility, will be able to execute the action with speed because of their operational agility and architect an innovative multichannel distribution through their partnering agility. Finally, the complexity of competitive actions the company is able to perform is increased thus creating competitive advantage.

Interestingly, EAM has been identified to foster organizational agility as well. By conducting case studies with four medium-sized companies, Fallmyr and Bygstad (2014) found that EAM is actually increasing organizational agility, in particular the capability to respond to external changes. Thereby, the authors disagree with concerns about EA as a new IT bureaucracy producing mountains of technically oriented documentations, which are very difficult to handle

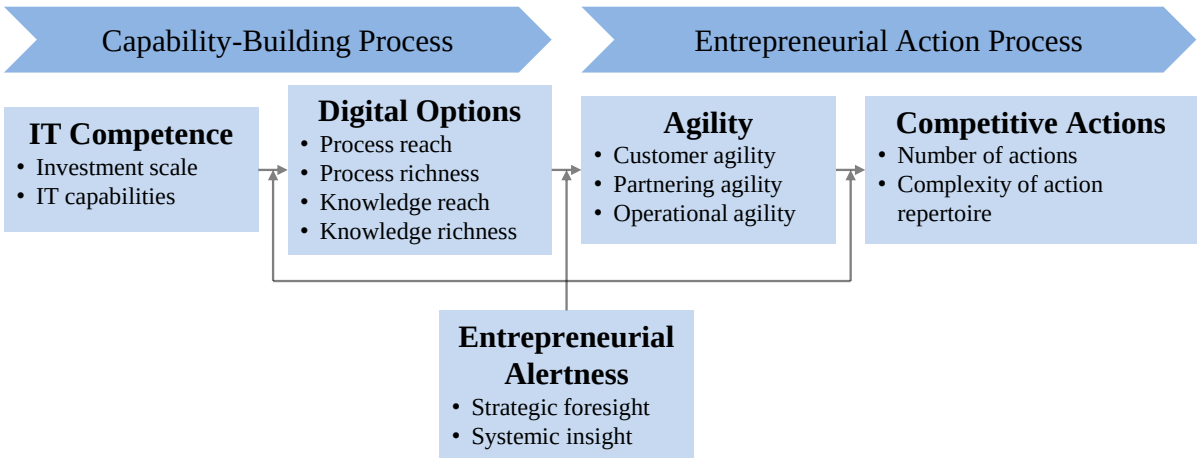


Figure 2.8.: Capability-building and entrepreneurial action based on on digital options according to Sambamurthy et al. (2003)

and keep consistent with the evolving enterprise. Most often, business-IT alignment is a major goal of any EAM initiative (see Chapter 2.2). Hence, it is important to demarcate alignment and agility. As Tallon and Pinsonneault (2011) point out, "firms with similar levels of alignment between their IT and business strategy could have different agility depending on their respective IT infrastructure flexibility".

It can be stated that alignment targets at internal coordination between business and IT departments whereas agility tries to integrate both in order to cope with a changing business environment. Nevertheless, ambidexterity—the capability of performing both exploration and exploitation in parallel—is required to "ensure short-term IT contributions and continuous progress of IT projects while simultaneously working toward IT transformation program success as a foundation for IT-enabled business transformation" (Gregory et al., 2015), especially during IT platform design, i.e. balancing IT standardization versus IT differentiation.

2.4. Complexity-based approaches to EAM

Over the recent years, the complexity view has been adopted by several researches active in the EAM discipline. For example, Vessey and Ward (2013) apply the complexity theory worldview which conceives of organizations and IS as complex adaptive systems that co-evolve over time, to develop a theory of sustainable IS alignment. Two relevant books summarizing many of the different approaches have been edited by Goetze and Jenssen-Waud (2013) and Saha (2014). Their analysis reveals that the different notions of complexity prevalent in general literature (see Chapter 2.4.1) are also found within the EAM literature. To better analyze related work relevant for this thesis and to structure it, a conceptual framework describing different notions of complexity is developed. Subsequently, this framework is applied to structure related work identified by following the guidelines for structured literature reviews (Webster and Watson, 2002; vom Brocke et al., 2009).

2.4.1. A framework based on identified complexity notions

In Chapter 2.1.2 four different dimensions of complexity notions prevalent in literature have been identified. First, the **organized vs. disorganized** notion presented by Weaver (1948) distinguishes complex problems based on the applicability of statistics. Second, the **qualitative vs. quantitative** notions can be distinguished based on the point of view whether complexity can be measured (Kolmogorov, 1963) or is a characteristic of a system summarizing phenomena like emergence (see Anderson et al., 2002) and self-organization (see Kauffman, 1993). Third, the **subjective vs. objective** dimension differentiates whether complexity is regarded to be a characteristic of the relationship between a system and its observer (Rosen, 1977) or the system itself (see Fioretti, 1999). Fourth, **structural vs. dynamic** notions of complexity can be distinguished. While the structure of a system can be complex (see Kolmogorov, 1963) the same holds true for its behavior (Sterman, 2000).

In this chapter, the independence of these dimensions is demonstrated by providing different examples from well-known literature in which different complexity notions are combined in various ways. Furthermore, a framework integrating all four orthogonal dimensions is presented. The framework will be used to explicitly document complexity notions used throughout this thesis.

2.4.1.1. Independence of complexity notion dimensions

Stephan Wolfram (1994) became famous for his groundbreaking research on one-dimensional cellular automata. Thereby, a cellular automaton consists of a regular grid of cells where each cell can have one of two states, i.e. *on* or *off*. Based on a given start configuration, i.e. specific states of the first line of the grid, a fixed rule is applied to calculate the states of the cells in the next line of the grid—a new generation. Such rule is applied to each cell individually and takes the states of the cell's neighborhood (typically its adjacent cells) into account. For example, rule 110 is described in Table 2.1.

current pattern	111	110	101	100	011	010	001	000
new state for center cell	0	1	1	0	1	1	1	0

Table 2.1.: Rule 110 for Wolfram's cellular automata

Based on their dynamic behavior Wolfram (1984) classified cellular automata into four distinct classes:

- Class 1: Almost all initial configurations relax after a transient period to the same fixed configuration.
- Class 2: Almost all initial configurations relax after a transient period to some fixed point or some periodic cycle of configurations, but which one depends on the initial configuration.
- Class 3: Almost all initial configurations relax after a transient period to chaotic behavior.
- Class 4: Some initial configurations result in complex localized structures, sometimes long-lived.

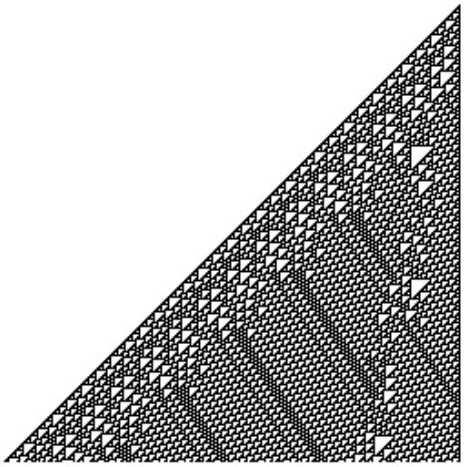


Figure 2.9.: Cellular automaton with rule 110 and 250 iterations

The most interesting class is Class 4, because Wolfram claimed that cellular automata belonging to this class can form universal Turing machines. The property of universality has at least been proven for Rule 110 (Cook, 2004) which was presented above and is visualized in Figure 2.9[1].

As already outlined by Schneider et al. (2014), for his classification of cellular automata, Wolfram combined the notion of **dynamic complexity** with the notion of **objective complexity** since individuals are not involved in the classification process. It is solely based on the patterns created by a cellular automaton. Since the four classes form a nominal scale rather than an ordinal scale he also applied the notion of **qualitative complexity** instead of measuring the complexity of cellular automata. Finally, statistical methods are not used to describe complexity of cellular automata so the notion of **ordered complexity** is applied.

Sterman (2000) applied another combination of complexity notions although analyzing—like Wolfram—the dynamic behavior of systems. With several experiments, Sterman analyzes how people understand the behavior of complex systems. For example, the Beer Distribution Game can be used to demonstrate how far from optimum people's decisions are in a relatively simple game. In this game subjects seek to minimize costs as they manage the production and distribution of a commodity. Though simplified compared to real firms, the task is dynamically complex because it includes multiple feedbacks, time delays, nonlinearities, and accumulations (Sterman, 1994). Therefore, Sterman (1994) concludes that the mental models people use to guide their decisions are dynamically deficient because they "adopt an event-based, open-loop view of causality, ignore feedback processes, fail to appreciate time delays between action and response and in the reporting of information, do not understand stocks and flows, and are insensitive to nonlinearities that may alter the strengths of different feedback loops as a system evolves".

When people learn in complex systems, Sterman (1994) distinguishes two types of learning. First, during single-loop learning "decision makers compare quantitative and qualitative information about the state of the real world to various goals, perceive discrepancies between desired and

[1] http://mathworld.wolfram.com/Rule110.html

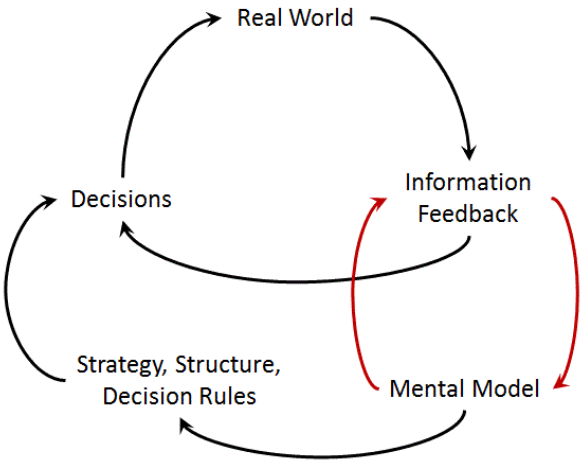


Figure 2.10.: Double-loop learning in complex systems according to Sterman (1994)

actual states, and take actions that (they believe will) cause the real world to move toward the desired state". Based on the idea of Forrester (1961), decisions are the result of applying a decision rule or policy to information about the world as we perceive it. Therefore, double-loop learning—as visualized in Figure 2.10—is more promising because it allows people to alter one's worldview.

We can easily see that Sterman (2000) considers **subjective complexity** as well as **dynamic complexity**. By assuming that there are different degrees to which people are able to understand complex systems, for example, if they perform double-loop learning, and that quantitative models of complex systems can be developed he also applies a notion of **quantitative complexity** although statistics might not be useful, i.e. **ordered complexity**.

By comparing the different notions of complexity being used by Wolfram (1994) and Sterman (2000) it becomes apparent that the notion of dynamic complexity can be combined with the notion of objective complexity (see Wolfram, 1994) as well as subjective complexity (see Sterman, 2000). The same holds true for qualitative and quantitative complexity although both authors apply a notion of ordered complexity. Accordingly, the question arises if a similar diverse combination of complexity notions can be found in the area of structural complexity.

Milgram (1967), for example, analyzed the structure of complex networks like social graphs and came up with the famous small-world property. Such networks have some very appealing properties and can be found in different areas. For example, the neural network of the worm Caenorhabditis elegans, the power grid of the western United States and the collaboration graph of film actors form such small-world networks (Watts and Strogatz, 1998). These networks display enhanced signal-propagation speed, computational power and synchronizability. Figure 2.11 shows how small-world networks reside somewhere between order and randomness.

As visualized in Figure 2.11, in small-world networks vertices are mostly connected to their neighbors but some of them are also connected to vertices far away. Thereby, small-world networks are highly clustered like a regular graph, yet with small characteristic path length, like a random graph.

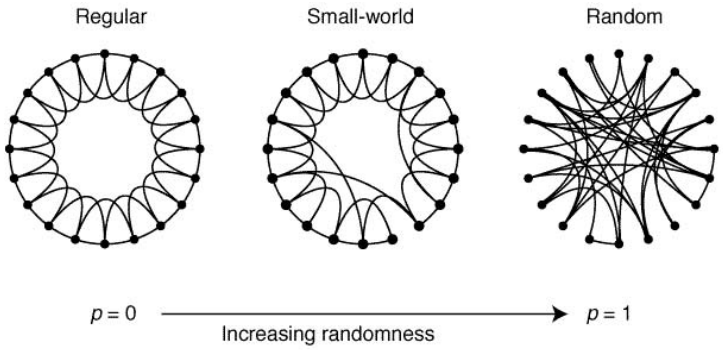


Figure 2.11.: Small-world networks according to Watts and Strogatz (1998)

Obviously, the authors apply an **objective complexity** notion since networks are analyzed without regarding the relationship with its observer. The authors apply a notion of **qualitative complexity** since their goal is not to measure the complexity of networks like social graphs but to explain why complex networks exhibit, for example, increased signal-propagation speed like in the experiment conducted by Milgram (1967) in which the median was only six hops in the social graph of arbitrary people who tried to deliver a letter to a randomly chosen person.

In contrast, Frese (1987) developed the idea of comprehensive complexity describing the relationship between a software user and a software system. In this context, complexity refers to decision necessities which should be optimized. The author also argues that control over one's activities should be increased at the expense of other features, e.g. complexity. Here, complexity is measured in terms of the number of goals, plans and signals as well as the number of their relationships between those and especially the number of conditional relationships. This allows to distinguish between a novice and an expert because although the number of elements stays the same the expert has to make less decisions than the novice thus experiencing less complexity.

Frese (1987) applies a notion of **subjective complexity** due to the fact that is regarded as "a characteristic of the interaction of the person with the environment" respectively the software system. By counting the number of different elements which increase the perceived complexity, he also applies a notion of **quantitative** and **structural complexity**. Thereby, statistical methods might be useful, thus the notion of **disorganized complexity** is present.

By comparing the different approaches of Milgram (1967) and Frese (1987), it can be stated that they combine different notions of complexity although both focus on **structural complexity**. While Frese (1987) defines complexity as a property of the relationship between a software user and a software system, thus having a notion of **subjective complexity**, Milgram (1967) applies a notion of **objective complexity** when he classifies networks. Likewise, the authors differ in their consideration of measures. Small-world networks form a class of networks based on the structure of their edges (**qualitative complexity**) but comprehensive complexity can be measured (**quantitative complexity**). Nevertheless, both authors focus on **disorganized complexity** because statistical methods are used.

The analysis of the complexity notions applied during the research activities conducted by Wolfram (1994); Sterman (2000); Milgram (1967) and Frese (1987) demonstrates the independence

	Structure vs. Dynamics	**Subjective vs. Objective**	**Qualitative vs. Quantitative**	**Organized vs. Disorganized**
Wolfram (1994)	Dynamics	Objective	Qualitative	Organized
Sterman (2000)	Dynamics	Subjective	Quantitative	Organized
Milgram (1967)	Structure	Objective	Qualitative	Disorganized
Frese (1987)	Structure	Subjective	Quantitative	Disorganized

Table 2.2.: Analysis of complexity notions in literature

of the complexity notions introduced in Chapter 2.1.2. Table 2.2 summarizes respective notions of complexity applied by the different authors. It shows that within the dynamic notion arbitrary combinations within the other dimensions are possible. Likewise, the same holds true for the structural notion of complexity. The objective as well as the subjective notion of complexity are combined with various manifestations of the other dimensions. Consequently, it can be concluded that any combination of complexity notions seems to be possible.

So far, it has been shown that in well-known complexity literature at least four distinct dimensions of different notions of complexity can be distinguished. By the examples given above, it has also been demonstrated that these dimensions can be combined in different ways. Based on this foundation, a conceptual framework for classifying complexity literature is developed in the next chapter complemented by a simple notation. This framework will then be used to classify existing work on complexity and therefore diversity in the context of enterprise architecture management.

2.4.1.2. A visual vehicle and notation

Based on the insight that in established literature at least four different dimensions of complexity notions can be distinguished (resting on the role of the observer, time, measures and statistical methods), all works addressing complexity or complex systems could be classified accordingly. Given the fact that these four dimensions seem to be independent a respective framework consisting of a visual vehicle and a simple notation can be developed. The visual vehicle is shown in Figure 2.12.

In this framework the horizontal axis is used to distinguish between the objective and subjective notion—each represented by a single small cube. The vertical axis is used to distinguish between the notions of structural and dynamic complexity. The qualitative-quantitative distinction is indicated by the third dimension of the cube. To visualize the fourth dimension distinguishing between ordered and disordered complexity different colors are used within the cube. The chosen representation clearly shows which notions of complexity exist and how the different notions can be combined. It has to be pointed out that a clear separation of notions within the same dimension is not always possible and that there can be a grey area in between. Furthermore, it is also possible to apply, for example, both a quantitative and a qualitative notion. Therefore, the conceptual framework serves as a basis for categorization not for detailed complexity notion descriptions.

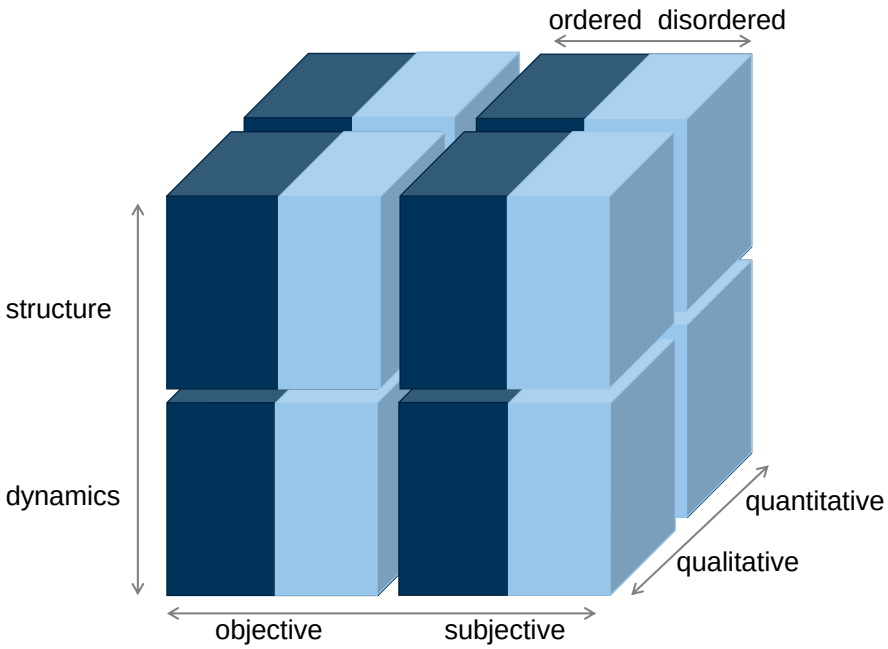


Figure 2.12.: The Complexity Cube: A framework unifying different notions of complexity based on Schneider et al. (2014)

To explicitly document the applied notions of complexity, for example, for a single publication, an understandable notation can be used (Schneider et al., 2014). The purpose of such notation is to facilitate literature research activities because similar works can be identified more easily and grouped into clusters. Since the proposed framework (see Figure 2.12) consists of four dimensions, a quadruple notation is proposed. To define such notation, four sets—each covering the notions of one dimension—need to be introduced first to form the foundation:

$$\begin{aligned} D_1 &:= \{objective, subjective\} \\ D_2 &:= \{structural, dynamic\} \\ D_3 &:= \{quantitative, qualitative\} \\ D_4 &:= \{ordered, disordered\} \end{aligned} \tag{2.1}$$

Given the four sets introduced in Equation 2.1, the set of applied complexity notions (ACN) can be defined as a quadruple as follows:

$$ACN := (x_1, x_2, x_3, x_4), x_1 \subseteq D_1, x_2 \subseteq D_2, x_3 \subseteq D_3, x_4 \subseteq D_4, \forall x_i, 1 \leq i \leq 4, x_i \neq \emptyset \tag{2.2}$$

Therein, each element x_i of the quadruple is a subset of one of the previously defined dimensions to categorize notions of complexity. Furthermore, these elements cannot be the empty set because one aspect of each dimension can be and needs to be chosen. As stated in the following

example, this notation can be used to document the applied notions of complexity for a single publication. Because the elements of the quadruple (x_i) are defined as subsets of a dimension—and not as strict subset—it allows to assign both poles of a dimension in case this is necessary.

Example 2.5: Application of the ACN notation. The work of Sterman (2000) has previously been classified to use the following notions of complexity: subjective, dynamic, quantitative, organized. Therefore, the ACN quadruple looks like follows:

$$ACN_{Sterman} := (\{subjective\}, \{dynamic\}, \{quantitative\}, \{organized\})$$

On the basis of the conceptual framework for different notions of complexity and the simple notation to document applied complexity notions introduced in this chapter, the next chapter summarizes related work on complexity in the EAM domain and analyzes prevalent notions of complexity.

2.4.2. An overview about EA complexity literature

Within the domain of Enterprise Architecture Management (EAM), insights gained by general complexity science have already been used to advance the field. To provide an overview about existing literature dealing with the concept of complexity in the EAM field and to lay down the foundation for the detailed diversity considerations in Chapter 4, a literature review is performed and described in this chapter. The results are presented and analyzed by the application of the complexity framework developed in the previous chapter and thereby extend the literature review conducted by Schuetz et al. (2013) who focused on complexity in the IS field in general.

2.4.2.1. Literature review process

As outlined by vom Brocke et al. (2009), "science is a cumulative endeavour as new knowledge is often created in the process of interpreting and combining existing knowledge". Therefore, the existing literature in the domain of EAM, in which the concept of complexity is used, is analyzed (see also Schneider et al., 2014). Thereby, the common guidelines for literature reviews presented by Webster and Watson (2002) have been used to guide the process which is visualized in Figure 2.13.

In the first step, the books of Goetze and Jenssen-Waud (2013) as well as Saha (2014)—both dealing with complexity in the field of EAM—have been analyzed to get an overview about existing works on this topic as well as used terminology and authors active in this field. Based on these insights, the EBSCOhost database, Science Direct, ISI Web of Science and the search engines of ACM, IEEE and Google scholar have been consulted in January and February 2014 to search for relevant publications. As search terms the following keywords have been used to query

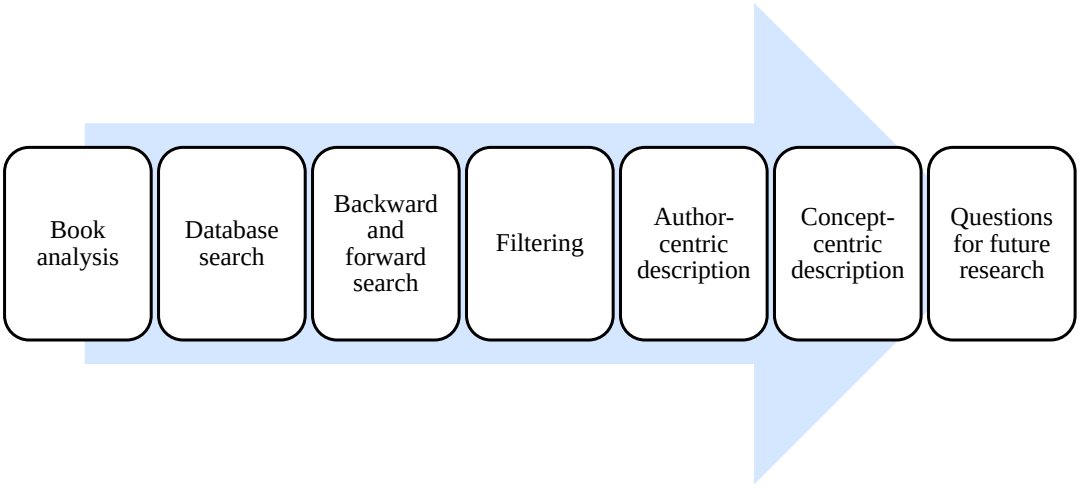


Figure 2.13.: Literature review process for EAM complexity publications

these databases for titles and abstracts: *"enterprise architecture" AND "complex*"*. Due to the frequent usage of these terms, the result list had to be filtered for publications dealing with the concept of complexity in the first place. After filtering ten particularly relevant articles have been identified. Although a backward as well as a forward search has been conducted on that basis no additional publications could be identified which are relevant in this context. The results are presented by an author-centric approach first (see Chapter 2.4.2.2) to provide a chronological overview. As suggested by Webster and Watson (2002), the results of this literature review are then also presented in a concept-centric way (see Chapter 2.4.2.3). Thereby, the concepts, i.e. dimensions of complexity notions, previously introduced are used to create a concept matrix. Based on this analysis questions for future inquiry are derived.

2.4.2.2. Literature review results

In this chapter, the results of the previously described literature review are presented by an author-centric approach as carried out by Schneider et al. (2014). Figure 2.14 provides a chronological overview of the identified articles.

Janssen and Kuk (2006) regard enterprises as complex adaptive systems (see Chapter 2.1.2) and attribute them properties like emergence and self-organization in order to derive requirements for a suitable EAM function. In addition, the authors provide concrete architectural guidelines which have been used to design an EAM function for a governance agency. These include, for example, that *diversity is important but should be created with care* and that *targets should not be set without constraints*. These guidelines target at both the enterprise's structure and dynamics. Furthermore, the authors do not consider the system's observer nor use statistical methods. Consequently, the complexity notions applied in the work of Janssen and Kuk (2006) can be classified as $ACN_{Janssen2006} = (\{qualitative\}, \{structural, dynamic\}, \{objective\}, \{ordered\})$.

Buckl et al. (2009) applied a complex adaptive system view on enterprises and used Beer's VSM (see Chapter 2.1.2) to derive and classify different duties of an Enterprise Architecture

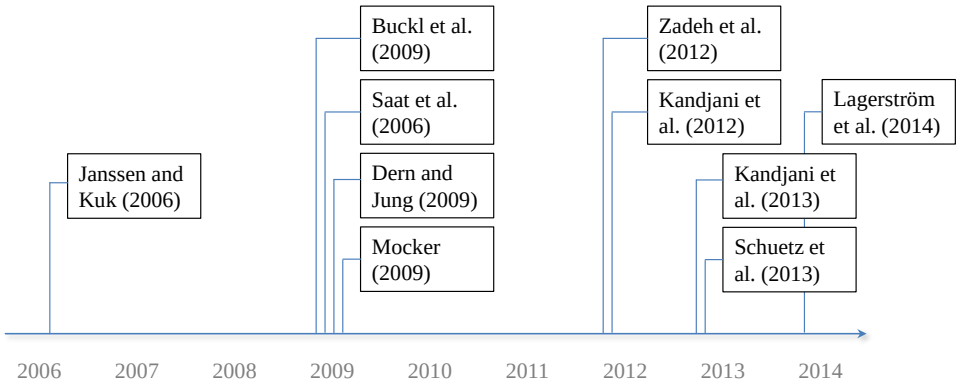


Figure 2.14.: Timeline showing literature review results

Management (EAM) function. Thereby, the authors distinguish between reactive and proactive tasks as well as EAM governance. Reactive tasks include setting up a structure, e.g. an intranet, where standards can be viewed and communicated to respective stakeholders. In contrast, proactive tasks are concerned with a holistic and future-oriented perspective to support strategic decision making. This includes, for example, EA analysis and target EA development as well as transformation planning. EA governance tasks are performed by system five according to the VSM and include the adoption of the overall management function and the provision of guidance to maintain the identity, i.e. the purpose of the EAM system. By using a well-accepted cybernetic model the authors are able to provide a framework to categorize EAM tasks in order to assess the completeness of existing EAM approaches. The authors have not attempted to measure how complex an organization is, focused on the structure of each VSM system, classified the enterprise as a complex adaptive system and focused on a manageable number of variables. Consequently, their complexity notion can be expressed as $ACN_{Buckl2009} = (\{qualitative\}, \{structural\}, \{objective\}, \{ordered\})$.

Saat et al. (2009) derive requirements for the design of an EA planning activity based on chaos theory. By attributing properties like sensitivity to initial conditions to the enterprise and therefore its architecture the authors clearly apply a qualitative notion of complexity here. Based on such attributions implications are derived. For example, different change volatilities and lifecycles of architectural elements may decrease the predictability of to-be models and therefore add complexity to EA planning. Furthermore, the authors identify structural invariance at different scales, for example, recurring structures and methods for planning the entire EA and or parts of it or a single architectural element (e.g. business application) and parts of it (e.g. components). In contrast to these structural aspects an enterprise's attraction to specific configurations is also assessed. Given the decreasing predictability for the suitability of to-be models with increasing temporal planning scope, multiple alternative versions of to-be models are developed. This enables enterprise architects to know several possible trajectories for the EA although it is not entirely predictable which trajectory the organization will chose. Obviously, the observer of the system is not part of the authors' considerations. Due to the fact that they do not use statistical methods, they apply an organized notion of complexity. Consequently, this work can be classified as $ACN_{Saat2009} = (\{qualitative\}, \{structural, dynamic\}, \{objective\}, \{ordered\})$.

Dern and Jung (2009) describe an IT architecture governance approach based on complexity measures. These measures focus on the structural complexity of the EA by considering the number of IT systems, their information exchange relationships and their homogeneity. The architect or any other observer is not part of their considerations. Similar to other quantitative approaches, their proposed metric for IT complexity aggregates numbers and calculates ratios, respectively. Consequently, their work can be classified as $ACN_{Dern2009} = (\{quantitative\}, \{structural\}, \{objective\}, \{disordered\})$.

Mocker (2009) provides an empirical evaluation of different EA complexity measures. The measures used to quantify the application architecture complexity include, for example, interdependencies of applications, diversity of technologies, deviation from standard technologies and redundancy. Thus, a quantitative as well as structural notion of complexity is applied. Obviously, the observer of the architecture is not part of these measures and statistics play a major role, implying $ACN_{Mocker2009} = (\{quantitative\}, \{structural\}, \{objective\}, \{disordered\})$.

Zadeh et al. (2012) explore whether the principles of cybernetics can provide a theoretical basis for interpreting EA principles derived through practice. Therefore, the authors use the Viable System Model (VSM) as well as its application to IT governance, the Viable Governance Model (VGM), to demonstrate how the architecture principles of The Open Group Architecture Framework (TOGAF) (see The Open Group, 2011) relate to those cybernetic concepts. The authors show that the nine business principles of TOGAF can be mapped to well-known cybernetic concepts like viability, recursion, cohesion, coordination and homeostasis. For instance, the TOGAF principle *Maximize Benefit to the Enterprise* is mapped to the cybernetic principles *viability* and *cohesion*. Beside applying this qualitative notion the authors also apply a quantitative notion due to their reliance on Ashby's law of requisite variety (see Ashby, 1956). The principles analyzed by the authors are mainly concerned with the structure of the enterprise architecture and not its dynamics while focusing on a manageable amount of variables. Consequently, their work can be classified as $ACN_{Zadeh2012} = (\{qualitative, quantitative\}, \{structural\}, \{objective\}, \{ordered\})$.

Kandjani et al. (2012) use the concept of EA cybernetics to determine the complexity of global software development projects. Thereby, the authors drop the idea that such projects are designed systems because deliberate design and control episodes are intermixed with emergent change episodes. To measure complexity the authors use three different indicators, e.g., by the number of relevant states of a system's environment. Furthermore, the axioms of extended axiomatic design theory are used to decrease the complexity of global software development projects. The authors conclude that life cycle activities of such projects should be made as independent, controlled and uncoupled as possible so that the designer can predict the next relevant states of the life history and avoid a chaotic change. To minimize information content of planning projects the management office should include the critical members from global software development companies with the most tacit knowledge of their companies. The used complexity metrics are independent of any system observer and by asking for the independence axiom within projects, the authors encourage a decoupled design focusing the projects structure. Finally, statistical methods are not part of their approach which results in $ACN_{Kandjani2012} = (\{quantitative\}, \{structural\}, \{objective\}, \{ordered\})$.

Kandjani et al. (2013) present a co-evolution path model which is based on the idea of Ashby's law of requisite variety (see Ashby, 1956). This model shows that each time the complexity of an enterprise's environment changes, the enterprise itself has to adjust its complexity accordingly. Because this adjustment, i.e. co-evolution, is a dynamic process, it is unlikely that the enterprise will exactly end up with the required complexity. Therefore, the model describes the path the enterprise's complexity will take along the optimal, i.e. required, complexity. This path is formed by the desired complexity, the desired excess complexity which is needed to achieve the desired complexity and the edge of chaos or undesired excessive complexity. In the latter state, the controller has to decrease the complexity to ensure enterprise viability. Because the system observer has no influence on the complexity calculations the complexity notions applied in this work can be formalized as $ACN_{Kandjani2013} = (\{qualitative, quantitative\}, \{dynamic\}, \{objective\}, \{ordered\})$.

Schuetz et al. (2013) present an approach to measure the structural complexity of EAs. Based on the framework developed by Schneberger and McLean (2003), the authors define the complexity of an EA as the number and heterogeneity of its elements and their relationships. To measure the amount elements are counted. To measure the heterogeneity Shannon entropy (see Shannon, 1948) has been selected due to several appealing properties in this context, for example, regarding additional characteristics, shifts in the classification, effects of small sets of elements and effects of proportional changes. In this approach, an individual system observer is not considered and the use of statistical methods is obvious resulting in $ACN_{Schuetz2013} = (\{quantitative\}, \{structural\}, \{objective\}, \{disordered\})$.

Lagerström et al. (2014) applied a concept well known in the software architecture domain, namely Design Structure Matrices (see Baldwin et al., 2013), to reveal the hidden structure of an application landscape. Therein, the authors model the application landscape as graph in which business applications form the nodes and their (reverse) information flows form the edges because they are interpreted as dependencies between business applications. Based on this graph, the overall architecture can be classified either as core-periphery, multi-core or hierarchical. Experiments conducted by the authors found only application landscapes which have been classified as core-periphery so far. In such setting, each business application is assigned to one out of four categories based on the number of their transitive dependencies: core, control, shared or periphery. The core applications form the largest cyclic group within the graph. Control applications depend on that core while the core itself depends on the shared applications. Periphery applications are only loosely coupled to the core. In addition, the so called propagation cost metric can be calculated. It is defined as the density of the visibility matrix (an adjacency matrix with transitive edges). Intuitively, propagation cost equals the fraction of the architecture affected when a change is made to a randomly selected element. For example, a propagation cost of 25% indicates that one-fourth of the architecture may be affected when a change is made to a randomly selected software application. Thereby, a concrete observer of the architecture is not considered implying $ACN_{Lagerstroem2013} = (\{quantitative\}, \{structural\}, \{objective\}, \{disordered\})$.

2.4.2.3. Analyzing prevalent complexity notions in EA research

In this chapter, the previously identified work related to complexity in the domain of Enterprise Architecture Management (EAM) will be presented in a concept-centric way as recommended by Webster and Watson (2002). The concepts used to structure the related work are provided by the complexity framework presented in Chapter 2.4.1. Based on the ACN-quadruples already defined for each work, Table 2.3 summarizes which notions of complexity have been applied.

Related work	**ACN D_1**	**ACN D_2**	**ACN D_3**	**ACN D_4**
Janssen and Kuk (2006)	qualitative	structural, dynamic	objective	organized
Buckl et al. (2009)	qualitative	structural	objective	organized
Saat et al. (2009)	qualitative	structural, dynamic	objective	organized
Dern and Jung (2009)	qualitative	structural	objective	disorganized
Mocker (2009)	qualitative	structural	objective	disorganized
Zadeh et al. (2012)	qualitative, quantitative	structural	objective	organized
Kandjani et al. (2012)	qualitative	structural	objective	organized
Kandjani et al. (2013)	qualitative, quantitative	dynamic	objective	organized
Schuetz et al. (2013)	quantitative	structural	objective	disorganized
Lagerström et al. (2014)	quantitative	structural	objective	disorganized

Table 2.3.: Overview of EA complexity publications classified by complexity notions based on Schneider et al. (2014)

When analyzing the chronologically ordered works listed in Table 2.3, a trend with respect to the applied notions of complexity in $ACN\ D_1$ can be observed. At the beginning the qualitative notion has been prevalent but after a while the quantitative notion became equally famous. This trend is reasonable because understanding the impact of complexity science on the field of EAM research qualitatively seems to be a prerequisite for measuring complexity. By analyzing $ACN\ D_2$, it can be seen that the dynamic notion of complexity is still underrepresented in current literature because most authors focused on structural complexity. The analysis of $ACN\ D_3$ reveals that all authors applied an objective notion of complexity. This implies that so far complexity is regarded to be a property of the system, i.e. EA, and not of the relationship between the architecture and its observer. Finally, the analysis of $ACN\ D_4$ shows that both organized and disorganized notions of complexity are represented equally. It can be concluded that future research should especially tackle the question how to describe dynamic as well as subjective complexity in the domain of EAM. In addition, the different complexity approaches published so far need to be complemented by additional experiments in real-world settings (see Schuetz et al. (2013); Lagerström et al. (2014)) and guidance for companies whether to increase or decrease their EA's complexity needs to be developed because it is missing up to now.

CHAPTER 3

Application landscape complexity and diversity perceptions in practice

"Change is accelerating, and as the complexity of the systems in which we live grows, so do the unanticipated side effects of human actions, further increasing complexity."

John D. Sterman, *Learning in and about complex systems*

As will be pointed out in this chapter, the diversity within an application landscape is a constituting aspect of its structural, objective and also subjective complexity. To begin with, causes and consequences of application landscape complexity as perceived by experienced practitioners are identified via focus group interviews. Then, the results of an online survey are presented indicating application landscape diversity (1) benefits and drawbacks expected and realized in practice, (2) steering mechanisms currently used in practice and (3) target values guiding management processes.

3.1. Application landscape complexity perception in practice

Given the various notions of complexity applied in scientific Enterprise Architecture Management (EAM) literature identified in the previous chapter, it is also of interest how practitioners today perceive application landscape complexity. Therefore, the results of focus group interviews are presented in this chapter. Foremost, the results include common drivers and consequences of application landscape complexity as perceived by practitioners.

3.1.1. Focus group interview approach

As described earlier, even among scientists there exists no agreed-upon definition for the term complexity (Johnson, 2007). To get one step further towards understanding complexity in the context of EAM and to complement approaches identified in literature, perceived drivers and consequences of application landscape complexity are identified in this chapter. The effects of complexity presented here are the results of exploratory focus group interviews (see Calder, 1977) with enterprise architects which have been conducted to gain insights about the following questions:

- What do today's enterprise architects perceive to be the key *drivers* of application landscape complexity?
- What do today's enterprise architects perceive to be the key *consequences* of application landscape complexity?

To answer these questions, a total of two consecutive focus group interviews was conducted, each lasting for half a day with a total of ten participants from eight different companies. Both focus groups were held in a metropolitan area in the south-east of Germany. The focus groups were formed in accordance with guidelines traditionally followed in the marketing research field (see Bellenger et al., 1976). Respondents were screened to ensure that they were currently or recently considered with application landscape complexity during their daily business. Only companies and respective architects have been invited which already actively try to manage or measure application landscape complexity. The resulting group was homogeneous with respect to job experience (all participants had more than ten years of relevant experience). Identities of participating firms were revealed to focus group participants to build trust and interest within the group. Industry branches represented in the group cover automotive, pharmaceuticals, banking, insurance and consulting. All companies are headquartered either in Germany or Switzerland but most of them operate on a global scale. The interviews were part of a broader research activity with this focus group which met five times between January 2013 and February 2014. The topic addressed by the group together with scientists from the Technische Universität München was *Complexity of Application Landscapes—Measures, Models, Management* (CALM^3). Questions asked by the moderator during interviews were mostly directly linked to the two main questions of the focus group interviews. If necessary, complementary questions regarding mentioned drivers and consequences have been asked to stimulate discussions. The approach is therefore consistent with procedures recommended for marketing theory development by several scholars (e.g. Deshpande, 1983; Bellenger et al., 1976). As described in the following, the responses of focus group participants about drivers and consequences of application landscape complexity were remarkably consistent across companies and industry sectors.

3.1.2. Drivers and consequences of application landscape complexity

The focus group participants mentioned the following *drivers* for application landscape complexity and achieved consensus about their impact. As indicated below, many of them are in line with previous research on application landscape complexity.

Local decision making Especially in federated or decentralized IT organizations, local decision making is prevalent. Local can have two meanings here. First, local can refer to the actual location of a business unit. For example, in the United States (US) decisions affecting the application landscape are not coordinated with departments in Europe. Second, local can refer to different business departments. For example, decisions affecting the application landscape made by the production department are not coordinated with decisions made by the sales department. This driver is in line with Ross et al. (2006) who identified *business silos* to be the first level of EA maturity.

Business complexity A driver setting a lower bound for application landscape complexity is the complexity of the business supported by the application landscape. For example, the more customer segments are targeted and the more sales channels and products are offered, the more complex the enabling IT will be. This driver for application landscape complexity is in line, for example, with Mocker and Ross (2013) who observed an increasing application landscape complexity at USAA after business complexity has been increased.

Legal requirements Legal requirements can drive the complexity of application landscapes in two different ways. First, new legal requirements force the implementation of new features. For example, Basel III[1] and Solvency II[2] require ad-hoc reporting of financial data which is usually realized by IT systems. Second, in some branches, e.g. financial services, authorities raise requirements regarding IT systems as well as their development and management processes. Such cross-functional requirements might increase the application landscape's complexity due to underspecification, lack of required knowledge to interpret these requirements and the need for involving many stakeholders as well as IT systems.

Technological progress Given the accelerating technological progress, the complexity of application landscapes increases. For example, current trends like "bring your own device" (see Ghosh et al., 2013) or continuous delivery (see Humble and Farley, 2010) impose changes to an existing application landscape. In addition, the exploitation of new paradigms such as in-memory databases (see Kemper and Neumann, 2011) requires tremendous change within an application landscape, thus increasing its complexity.

Short-term optimization If new business capabilities should be supported by IT usually there are two different ways: a "quick and dirty" way and an architecturally consistent way. This is, for example, also reflected by the managed evolution approach proposed by Murer et al. (2008) which tries to balance both opportunities in the long run. If the quick and dirty approach is chosen too often for short-term optimization, the complexity of the application landscape increases. Schmidt and Buxmann (2010) also reported that the solution scope of system changes is typically local and focused on short-term goals while Gregory et al. (2015) consistently observed an organizational tendency to drift toward short-term IT demands.

To complement these factors driving application landscape complexity, the focus group participants achieved consensus about the following *consequences* of application landscape complexity during the respective discussions:

[1] http://www.bis.org/publ/bcbs188.htm
[2] http://ec.europa.eu/finance/insurance/solvency/solvency2/index_de.htm

Costs An increase in application landscape complexity is considered to cause an increase in related costs. In this context, costs occur on various levels or phases which are all affected. These include, for example, operating costs, maintenance costs and development costs. Especially the interconnectedness of business applications can result in increasing project sizes which in turn are considered to increase costs. The same effect is attributed to increasing management efforts. The impact of complexity on costs is in line with the observations made by, e.g., Ashkenas (2007) and Mocker (2009).

Errors An increase in application landscape complexity results in an increase in errors during operations. To justify this consequence of application landscape complexity on the focus group participants calculated correlation coefficients for application landscape complexity (number of applications, number of interfaces, heterogeneity of technology components,...) and incidents. Thereby, a significant correlation between both could be observed. Although not having performed similar calculations, other participants agreed with this observation.

Skill dependency Complexity within an application landscape might vary, e.g., across functional domains. People such as architects, developers or business analysts considered with a single domain for many years are able to understand the respective complexity better than newcomers. If changes should be made with in a distinguished domain people understanding the IT systems in that domain are required. Therefore, an increase in application landscape complexity is considered to result in an increasing dependency on specific people and their skills.

Agility In general, application landscape complexity is associated with decreasing agility meaning an increased time-to-marked for new IT systems. This is in line, e.g., with the observations made by Ross et al. (2006, p. 11) and can be explained by an increase in interconnectedness of IT system and therefore propagation of changes (Lagerström et al., 2014). Nevertheless, this opinion of the focus group is in contradiction with digital options theory stating that an increase in IT capabilities provides more digital options which can be used to create agility (Sambamurthy et al., 2003). Thus, a detailed analysis of this application landscape consequence is required (see Chapter 3.2.3).

Shadow IT An increase in application landscape complexity is associated with an increase of shadow IT. Here, shadow IT does not only cover IT systems directly setup by business units without knowledge of the IT department, it also includes IT systems business units put in place because government structures do not prevent them to do so. The reason why business units might act like this could be due to increasing time-to-market if the central IT department is responsible for the introduction of a new system or expected cost savings. Thereby, this expectation is in line with previous research (Rentrop and Zimmermann, 2012).

During the discussions within the focus group interviews, the following drivers or consequences have also been mentioned: limited sourcing opportunities, increasing training period for new employees and missing modularity. Due to missing consensus within the group and partial overlap with other drivers or consequences these concepts have been excluded from further considerations.

It is important to note that there was consensus in the group that heterogeneity of application

landscape components is also a driver for application landscape complexity. Nevertheless, it has been decided that heterogeneity—and therefore diversity—is an inherent part of complexity (as is the amount of application landscape components) and can therefore be excluded from the drivers and consequences list. Figure 3.1 visualizes the main results of the focus group interviews approved by all participants. The input created by the focus group interviews is not intended to reflect how strongly the different relationships shown in Figure 3.1 are. Conclusions about the absolute or relative strength would require a quantitative approach in which strength of conviction is specifically measured.

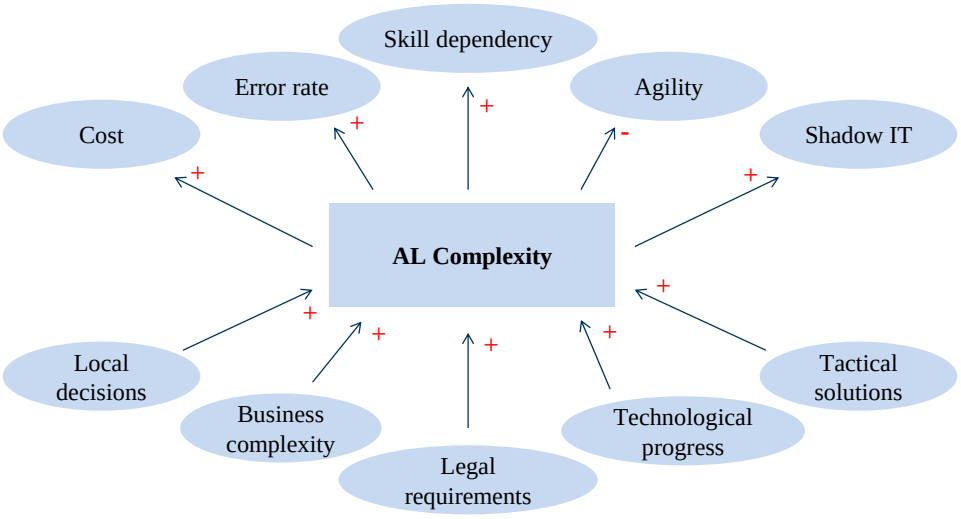


Figure 3.1.: Drivers and consequences of application landscape complexity as presented by Schneider and Matthes (2014b)

3.2. Application landscape diversity perceptions in practice

To complement the previously described perceptions of application landscape complexity in practice, this chapter focuses on perceptions about the effects of application landscape diversity. The goal is to analyze which benefits and drawbacks of standardization or differentiation can be realized in practice. Although a quantitative approach seems to be suitable to develop a detailed picture of today's diversity management approaches and their effects in practice, it would be difficult due to missing conceptualizations of both standardization and affected properties, e.g. agility or costs. In addition, as will be outlined in Chapter 5.3, suitable metrics to quantify variables required for quantitative analyzes are also missing. Therefore, the goal of the online survey presented in this chapter is to make the first step towards a broader understanding of today's application landscape diversity management approaches in practice. Beside the analysis of goals pursued by diversity management and respective means, perceived effects of application landscape diversity for which either contradictory assertions are found in literature or empirical evidence is scarce, are inquired.

3.2.1. Theoretical foundation and propositions

In this chapter, existing theories regarding the effects of application landscape diversity are reviewed. Based on this foundation several questions are derived which will be addressed by an online survey described in the following chapter.

Diversity's impact on costs

Although research on the management, governance and enforcement of IT standards within organizations is relatively limited (van Wessel et al., 2005), increasing costs are frequently attributed to application landscape diversity which is in line with the micro economic principle of *economies of scale*. Table 3.1 analyzes a selection of statements about the relationship between application landscape diversity or standardization and related costs. From this analysis one can see that different authors expect or observed an increase in related costs if the diversity within the application landscape increases. However, the cost concept used so far seems to be quite general because different kinds of costs are often not distinguished. Due to missing empirical evidence for cost reduction related to standardization (Boh and Yellin, 2007), it should be assessed whether practitioners are able to realize cost savings by means of reducing diversity of application landscapes.

Statement	**Source**
"[IT] centralization facilitates standardization, integration, and economies of scale."	DeSanctis and Jackson (1994)
"Research suggests that IT standardization can help to considerably reduce the overall IT costs of a company."	Mueller et al. (2015)
"Reducing the heterogeneity of infrastructure components would enable organizations to better leverage the investment in physical IT infrastructure across units."	Boh and Yellin (2007)
"[S]upporting a large number of different IT platforms complicates systems development and increases maintenance costs."	Boh and Yellin (2007)
"The new [standardized] infrastructure services introduced cost savings and new capabilities."	Ross et al. (2006, p. 38)
"Fewer [IT] platforms mean lower cost."	Ross et al. (2006, p. 74)
"The advantages of a homogenous business bus are [..] cost savings in implementing new and maintaining existing applications."	Puschmann and Alt (2004)

Table 3.1.: Selected propositions on application landscape diversity and costs

Diversity's impact on IT system integration

In literature, diversity is frequently considered to hamper interoperability among IT systems. Table 3.2 provides an overview of respective statements. One can see that standards here consider both IT components and data transfer protocols. Due to the fact that often different standards exist on the market or within an application landscape, it should be assessed if standardization fosters interoperability of IT systems and if different standards used in parallel hinder interoperability.

Statement	Source
"Standards help collaboration among vendors to provide integrated information system to the end-users. This helps to provide simplified IT infrastructure with seamless integration of divergent systems."	Kumbakara (2008)
"EA standards for physical IT infrastructure reduce the heterogeneity and increase the compatibility of IT infrastructure components across business units by limiting technology choices, reducing the number of platforms supported, and defining standard protocols to enable communication between systems."	Boh and Yellin (2007)
"Standards provide harmonized applications of IT products, facilitate the interchange of information, and provide appropriate degree of interworking amongst various systems."	Ngosi and Jenkins (1993)
"After all, an IT standard is not a means in itself, but a means to an end: interoperable applications."	Jakobs (2005, p. xiii)
"[T]he use of standards leads to the uniformity of objects and therefore enables and facilitates interaction between at least two objects."	Buxmann et al. (1999)

Table 3.2.: Selected propositions on application landscape diversity and interoperability

Diversity's impact on innovation

Previous research associates decreasing application landscape diversity, i.e. standardization, with an increase in innovation (Ross et al., 2006; Hui et al., 2013). Table 3.3 provides an overview of selected propositions. However, increasing application landscape diversity, i.e. differentiation, has also been identified as means to increase innovation capabilities (Gregory et al., 2015). Given these contradictory statements the question arises whether practitioners can realize more innovation by using standardized IT or more diverse IT.

Statement	Source
"[S]tandardized data and processes allow for process innovation closer to the customer."	Ross et al. (2006, p. 99)
"[Technical standards] form the basis for new technology and promote technical innovation through regulation of accumulated technical experience."	Hui et al. (2013)
"Usually a standard is a restriction in one layer that creates lots of possibilities, freedom, flexibility and incentives to innovation in other layers, either above or below the standard."	Vrancken (2006)
"Winning the support from business and Bank1's top management during year 5 of our case was enabled by synergistically blending IT efficiency with IT innovation and accommodating IT standardization needs with business demands for IT differentiation."	Gregory et al. (2015)

Table 3.3.: Selected propositions on application landscape diversity and innovation

Diversity's impact on agility

In addition to innovation, application landscape diversity is expected to influence an organization's agility. Therefore, Table 3.4 provides an overview of selected propositions. Again, one can see that literature yields inconsistent assertions about the influence of diversity on organizational agility. For example, according to Ross et al. (2006), a less diverse application landscape increases global flexibility whereas according to Wybo and Goodhue (1995), a less diverse application landscape can reduce flexibility of individual subunits. Therefore, the question arises how practitioners currently experience the impact of application landscape diversity regarding organizational flexibility.

Diversity's impact on manageability

As described earlier, application landscape design is today considered to be a main task of enterprise architects. Accordingly, the desired degree of diversity might be influenced by its impact on the manageability of the application landscape. Table 3.5 provides some selected statements about the impact of diversity on manageability. For example, according to Mueller et al. (2015), IT standardization represents a possible means to reduce (subjective) complexity and to ease application landscape management. Therefore, the question arises if practitioners actively decrease diversity within the application landscape to ease its management and maintain control.

Statement	Source
"[W]hen uncertainty comes mainly from task complexity or environmental instability unique to individual subunits [..] semantic standards can reduce the flexibility of an individual subunit to redesign its systems to best meet its unique information requirements."	Wybo and Goodhue (1995)
"[T]he use of standardized technologies increases global flexibility by reducing technical complexity and thus reducing implementation time."	Ross et al. (2006)
"Usually a standard is a restriction in one layer that creates lots of possibilities, freedom, flexibility and incentives to innovation in other layers, either above or below the standard."	Vrancken (2006)

Table 3.4.: Selected propositions on application landscape diversity and agility

Statement	Source
"[S]tandards are essential for companies to manage and maintain control over their IT."	Mueller et al. (2015)
"[S]etting and controlling a well-defined, robust set of IT standard are the foundation for effective IT governance."	Park et al. (2006)

Table 3.5.: Selected propositions on application landscape diversity and manageability

3.2.2. Survey design

To collect data required for assessing the current state of application landscape diversity management in practice regarding previously mentioned goals, an exploratory online survey has been designed and conducted as "initial research, which forms the basis of more conclusive research. It can even help in determining the research design, sampling methodology and data collection method" (Singh, 2007, p. 64). The following chapters describe the data collection method and provide an overview about the experts who participated in the survey.

Data collection method

As outlined above, assessing both application landscape diversity and its effects in practice and applying quantitative methods to prove some previously derived hypotheses is currently not possible due to missing concepts, metrics and data. Therefore, the survey described in this chapter only collects data about the perceptions of enterprise architects and software engineers regarding previously selected effects. In addition to answering structured questions, participants are also asked to comment on their answers and share their experience and mental models via open-ended questions. Thereby, a broad overview about currently applied diversity management approaches can be derived, recurring reasons or effects identified and misunderstandings avoided.

The survey was conducted by Schneider et al. (2015b) in the middle of 2015. Invitation links to the online survey were mailed to enterprise architects and software engineers from Germany, Austria and Switzerland who had previously collaborated in research endeavors with the chair or visited respective conferences. A detailed overview of respondents is given in the next chapter. The available alternatives to answer questions were predetermined by the researchers and generated using existing literature (see previous chapter) as well as the researchers' own experience. To assess whether application landscape diversity has an impact on certain other properties, common Likert scales were used. As with most research designs, the one described and used here has limitations. First, the number of responses is not as large as it would be expected, e.g., for a causal research survey. Second, the sample is not random which might bias the results although it remains unclear in which way. Third, only selected effects of standardization are assessed. However, it needs to be noted that "an exploratory study [as described here] may not have as rigorous a methodology as it is used in conclusive studies, and sample sizes may be smaller" (Nargundkar, 2003, p. 41).

Description of responses

In total, 47 practitioners correctly filled out and returned the initial survey while 24 answers were excluded because of missing answers or missing expertise. Due to the anonymous submission opportunity, it cannot be excluded that a certain company participated twice. However, all 37 participants for which their identities are known work for different companies or in two cases different branches within the same company. The companies that returned the survey were from a wide range of industries (see Figure 3.2) and different sizes (see Figure 3.3). However, most of them stem from the financial, technology and production sector and have more than 50,000 employees.

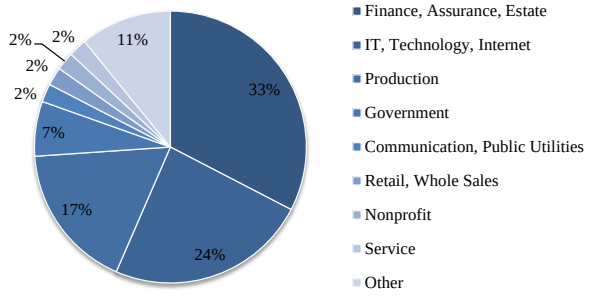


Figure 3.2.: Industry sectors of participants

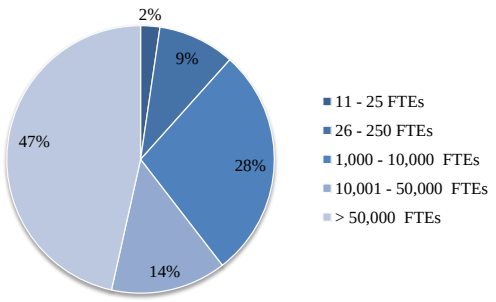


Figure 3.3.: Company sizes

The respondents are mostly enterprise architects (57%), followed by IT managers (15%), consultants (15%) and project managers (7%). 76% have more than nine years of professional experience. According to their own assessment, 26% of the respondents rated the maturity of their EAM function low, 50% medium and 24% high. In about one half of the participating companies the EAM initiative is manly IT driven, in the other cases it is both business and IT driven. In about three fourths of the participating companies (76%), the EAM initiative has been started five years ago or earlier. One third (31%) of the respective enterprises has started to manage their EA more than ten years ago. In 89% of these companies, the EAM initiative has

the support of the upper management. Of 43 respondents, 83% stated that IT standardization is a key challenge for their organization. Accordingly, only 15% of the respondents classify their current application landscape as highly standardized. 53% classify their degree of standardization as medium while 32% have a low degree of standardization, i.e. a high degree of diversity, within their application landscape. However, 89% confirmed that responding quickly to changes in the environment of the enterprise is either important or very important.

3.2.3. Data analysis

In this chapter, the analysis of the survey responses is described. First, the current state of diversity management in practice is assessed. Second, the impact of application landscape diversity on the previously described aspects is examined.

Current state of diversity management in practice

To assess the motivation behind application landscape diversity management in today's companies, the most prominent reasons for IT standardization are identified. When asked for the main reasons why their company tries to decrease the diversity of its application landscape, 85% of the respondents named *cost reduction*. This main goal is followed by *maintaining control and manageability* (66%), enhancing *integration and collaboration* (55%), implementing *security guidelines* (45%) and implementing *regulatory requirements* (26%). In addition, the enhancement of IT operations was mentioned because less diverse knowledge is needed. Interestingly, no value-adding goal has been mentioned at all.

Because there is no consistent knowledge available about how to standardize an application landscape, companies were asked about common means they implement to do so. Table 3.6 provides an overview about respective answers.

Means	Implemented	Not implemented
Definition of standard products	38	8
Definition of technical platforms	37	9
Requiring concessions for non-standard products	33	13
Definition of standard compliant target architecture	32	14
Definition of architecture principals	31	15
Transformation/migration of existing solutions	29	17

Table 3.6.: Currently implemented means for decreasing application landscape diversity

It is interesting to see that most means currently used to manage diversity target on guiding future decisions. The active migration to standard products is not as often used as other means. However, 75% of the respondents confirmed that only specific areas of the application landscape are subject to standardization. Therefore, it is not surprising that 63% of the respondents answered that, in the company they work for, application landscape diversity is actively increased in specific areas. These areas include, for example, non-core business areas like HR payroll and

areas in which distinction from competitors should be achieved. Therefore, as listed in Table 3.7 different means are currently used in practice to achieve an increase in application landscape diversity. The most popular means is to define multi-product strategies, e.g. for relational database management systems. In some cases diversification is also achieved be either reducing the influence of the central IT management division or by starting completely new business units. Respondents also mentioned that diversity is increased via extending the portfolio of standard products or via permitting more exceptions. However, 35% or the respondents were not aware of any means currently used to increase application landscape diversity.

Means	Implemented	Not implemented
Multi-product strategy	17	29
New subsidiaries or companies	6	40
Reduction of central management	6	40
Respective specifications	3	43

Table 3.7.: Currently implemented means for increasing application landscape diversity

Although application landscape design is a core task of enterprise architects, it remains unclear which leverage points qualify for starting standardization efforts and if enterprise architects are usually able to exert their influence on them. According to the participating companies, the various layers of an EA are not equally suitable for standardization (see Figure 3.4). For example, infrastructure elements are considered to be a better leverage point than technical components. Furthermore, business applications should be preferred to business processes whereas business objects or data is considered to be least useful for standardization. By looking at the distribution of the influence enterprise architects usually have on these layers, it becomes obvious that they cannot influence all suitable leverage points equally (see Figure 3.5). For example, enterprise architects have more influence on the standardization of data than on business processes although they are less suitable for standardization. Among the survey participants, 61% attributed high or very high relevance to business processes but only 16% have also high or very high impact on their design. Figure 3.5 additionally indicates that the current influence of enterprise architects is heavily IT focused and less business focused. Accordingly, enterprise architects might not be able to manage diversity within the application landscape alone due dependencies to other layers they cannot fully design.

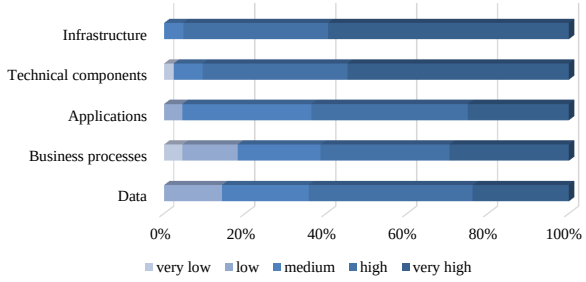


Figure 3.4.: Standardization leverage points

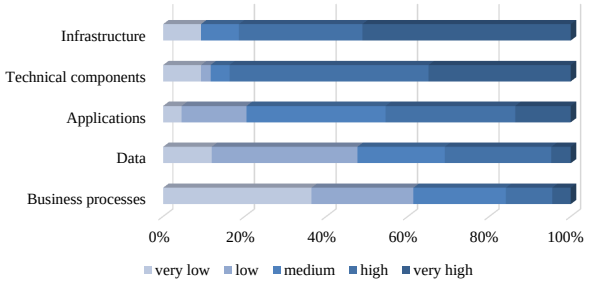


Figure 3.5.: Influence of enterprise architects

Benefits and drawbacks of application landscape diversity

As mentioned before, expected cost savings are the most prominent reason why the companies participating in this survey decrease the diversity of their application landscape. Accordingly, 81% of the respondents think that IT standardization reduces IT costs either significantly (41%) or slightly (40%). Interestingly, only 41% of these companies confirmed that respective cost assessments are actually carried out. Thus, it can be concluded that the unanimous opinion of practitioners today is that diversity is directly associated to costs although some critics can also be found. When asked about the expected amount of realizable cost savings, 25% expect savings less or equal to 5% of the overall IT budget. Only 13% expect a potential for savings greater than 20% of the IT budget. Thereby, the greatest potential for savings is attributed to operating costs (95%), followed by maintenance costs (86%) and costs for new IT developments (68%). However, 60% of the respondents could not confirm the statement that a more diverse application landscape produces more incidents during operation. Out of 30 respondents who answered the question about a specific case in which disregard of a standard yielded monetary losses 37% were able to describe such case. These cases included, for example, the development of redundant infrastructures (ESB, DWH) and the parallel development of different user interfaces.

In addition to saving costs, keeping the application landscape under control and reducing its complexity has been identified as the second most important reason why companies try to reduce the diversity of their application landscape. Although only 66% of the respondents named maintaining manageability as a reason for decreasing application landscape diversity, 93% confirmed that standardization is a suitable means to achieve it. Consequently, it can be regarded as a suitable means to reduce application landscape complexity. However, out of 38 respondents who answered a question about the quality of standardized solutions 74% think that they result in sub-optimal local solutions. Furthermore, 39% report that they have seen such behavior often or very often. An additional portion of 39% said that standardization yields sub-optimal local solutions at least sometimes leaving only 22% saying that less optimal solutions occur only rarely or never.

Regarding the hypothesized increase of interoperability due to decreasing diversity, 91% of the respondents were able to describe at least one case in which the usage of complementary standards was successful. Here, an interesting correlation can be observed. In 31% of the responding companies “many” such cases could be observed and only in 9% none could be observed. Every of the former companies uses a specific EAM framework while non of the latter companies uses such framework. In total, 67% of the respondents agreed that IT standardization fosters interoperability. However, only 8% of these companies actually assess the impact of standardization on interoperability. In addition, 62% of the respondents agreed that the existence of incompatible standards is prevalent and hinders integration of IT systems.

The impact of application landscape diversity on innovation is manifold. First, 70% of the respondents were able to describe at least one case in which standardization was fostering technological innovation. In addition, the same amount of respondents agreed that standardization on a specific layer can foster innovation on the next layer above. This includes, for example, company-wide unified communication solutions, single sign on solutions and consistent source code management through a single team server. The most frequently mentioned enhancements

due to standardization include data aggregation and business process optimization. However, this needs to be viewed in the light of the previously described limitations of standardized application landscapes regarding the fulfillment of (local) demands and the fact that 75% of the respondents confirmed (6% refused) that business departments expected to generate innovations require a higher degree of application landscape diversity.

63% of the respondents confirm that application landscape diversity impacts the agility of business departments. The respondents mention, for example, that standard solutions can be commissioned faster and therefore business units can use them earlier. Furthermore, 81% of the respondents confirmed that standardization results in technology or vendor dependence. 53% ranked this impact strong or very strong. Thus, agility can be limited if diversity is decreased.

In summary, the conducted survey confirmed not only the need for decision support in the context of application landscape diversity management but also revealed questions for future research. For example, cost savings are the most prominent reason why companies today try to standardize their application landscape. However, this comes with drawbacks such as less optimal business solutions and technology dependence. Up to now, it remains unclear which type of costs can be reduced (operation costs, maintenance costs or development costs) and to what extend this is actually achievable. In particular, there is no evidence that cost savings can be achieved via standardization while maintaining the same level of functionality. In future, a quantitative approach might yield more detailed insights here. Beside that, enterprise architects currently not always have the required influence within the organization to manage the diversity of the most important parts of the EA. In addition, application landscape diversity seems to be related to subjective complexity because standardization has been mentioned as a suitable means to maintain manageability of the application landscape. This aspect will be discussed in following chapter.

3.3. The role of diversity for application landscape complexity

So far, application landscape complexity has been discussed with regard to its drivers and consequences as they are perceived in practice (Chapter 3.1). In addition, the results of a survey assessing the perceived drivers and consequences of application landscape diversity have been presented (Chapter 3.2). Accordingly, this chapter outlines the theoretical assumptions about the relation of complexity and diversity in general and relates application landscape complexity and diversity by comparing the previously described observations.

3.3.1. Theoretical insights for systems in general

In complexity theory (see Holland, 1995; Kauffman, 1995), internal diversity is regarded as a fundamental requisite of systems to achieve a higher rate of adaptability. This is in line with Ashby (1956) who argues that a system has to embody as much variety as its environment. In addition to Ashby's law of requisite variety, Allen (2001) proposed the law of excess diversity, stating that a sustainable evolutionary strategy requires an amount of internal diversity superior to that of the environment. To conclude, diversity constitutes an important element in

short- and long-term adaptability (Andriani, 2001). These general observations should therefore also hold true for organizations which are frequently regarded as complex systems (Buckl et al., 2009; Ladyman et al., 2013). Self-organization, a system property attributed to complex systems (Ashby, 1947; Kauffman, 1993) also depends on diversity of system elements enabling autocatalysis (Juarrero, 2000). Therefore, it can be summarized that (1) adaptive strategies are more successful if agents and systems are internally diverse and (2) the transition from an aggregate to a system is also dependent upon internal diversity (Andriani, 2001).

Another research area in complex systems investigations is robustness (Jen, 2005). Although many definitions of robustness exist (Krakauer, 2006), it can be generally defined as the ability of a system to maintain functionality in the face of some change or disturbance. Thereby, robustness needs to be distinguished from stability which refers to the tendency of a system to return to an equilibrium after facing some change. In contrast, the concept of robustness covers continuous sustainability as well as responsiveness to shocks. Thus, it expands on the notion of resilience which considers the ability of a system to recover from trauma (Page, 2011). Organizations can achieve robustness by performing both exploration and exploitation (March, 1991). In order to explore, at least a temporal increase in diversity is required (Page, 2011). In ecology, for example, robustness is directly liked to diversity as it is considered to be a measure of the preservation of species diversity upon species removal (Krakauer, 2006).

Diversity can also have a stabilizing effect in complex systems. Kirman and Vriend (2001) present a simple market model in which individual buyers and sellers are allowed to strike up relationships with one another. If buyers differ in the price at which they value the goods, then buyers with relatively high values tend to pay higher prices. According to Page (2011), in this model diversity produces complexity through the web of connections and reputations that emerge from the system. Without diversity nothing interesting would happen. But with diversity market prices stabilize over time although the behavior of agents forms a complex system.

In the field of information systems, complexity is often regarded as number and variety of both computing components and component interactions (Schneberger and McLean, 2003). This concept has recently been transferred to the domain of enterprise architectures (Schuetz et al., 2013). Thereby, complexity has been defined as the number and heterogeneity of both EA elements and their interactions. Accordingly, heterogeneity, i.e. one aspect of diversity (see Chapter 4), seems to form an integral part of information systems and enterprise architecture complexity.

3.3.2. Empirical insights for application landscapes

Although theoretical propositions about the relationship of diversity and complexity exist, it should be assessed whether this relationship can also be proven empirically in the context of application landscapes. Therefore, in addition to the effects of application landscape diversity analyzed in Chapter 3.2.3, the respondents were also asked about the influence of application landscape diversity on the drivers and consequences identified for application landscape complexity in Chapter 3.1.2. The drivers for application landscape complexity identified via focus group interviews include *local decision making, business complexity, legal requirements, technological*

progress and *short-term optimization*. The consequences of application landscape complexity include *costs, errors, skill dependence, inflexibility* and *shadow IT*. Accordingly, the survey participants were asked if each of the identified drivers of application landscape complexity is also a driver for application landscape diversity. Table 3.8 summarizes the results.

Driver	(Strong) Confirmation	Indecision	(Strong) Refusal
Local decision making	92%	6%	2%
Business complexity	60%	23%	17%
Legal requirements	13%	9%	78%
Technological progress	83%	6%	11%
Short-term optimization	87%	9%	4%

Table 3.8.: Complexity drivers and their influence on diversity

The results indicate strong support for the proposition that the drivers of application landscape complexity are also drivers for its diversity with one exception, i.e. legal requirements. Although the increasing speed of new laws impacting application landscapes and their increasing scope drive the complexity of application landscapes, the opposite seems to be true for its diversity. According to the survey participants (78%), it becomes easier to comply to legal regulations if the application landscape is less diverse.

Likewise, the practitioners survey about effects of application landscape diversity was used to assess whether it yields the same consequences as application landscape complexity does. Table 3.9 summarizes the results.

Consequence	(Strong) Confirmation	Indecision	(Strong) Refusal
Costs	81%	17%	2%
Errors	41%	38%	21%
Skill dependence	78%	15%	7%
Inflexibility	37%	17%	46%
Shadow IT	57%	21%	22%

Table 3.9.: Complexity consequences influenced by diversity

As the survey results indicate, application landscape diversity has similar consequences as application landscape complexity has. This includes at least increasing costs and increasing skill dependence. An increase in errors during operation is also slightly confirmed. In contrast, 46% of the respondents see an increase in flexibility due to an increase in diversity. This is partly in line with the previously described analysis results regarding flexibility. Furthermore, there is support for the proposition that diversity does not increases shadow IT (57%) contrary to the observed effect of application landscape complexity. However, it can be concluded that also from an empirical point of view the concepts of complexity and diversity are quite similar in the context of application landscapes as they share many drivers and also consequences. Figure 3.6 summarizes these outcomes by indicating drivers and consequences similar as well as different (highlighted in green) to application landscape complexity.

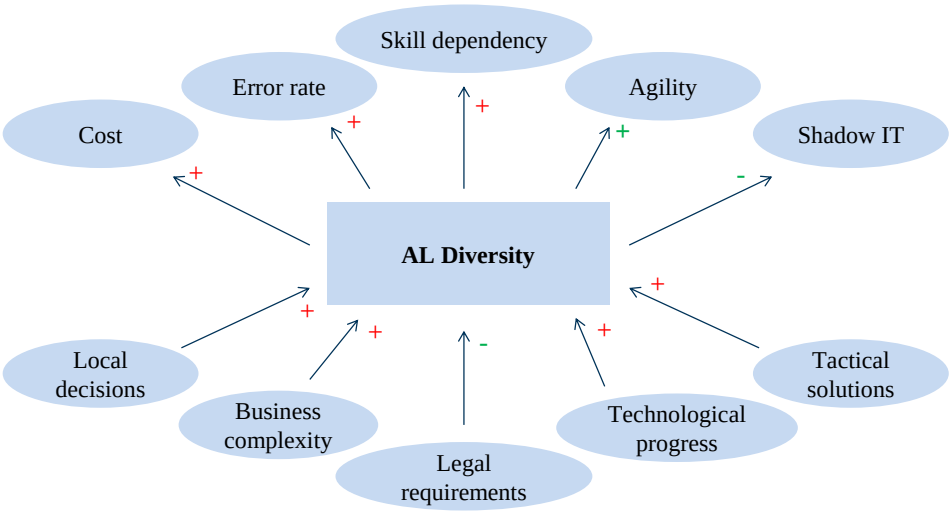


Figure 3.6.: Drivers and consequences of application landscape diversity

CHAPTER 4

A conceptual framework for application landscape diversity

"It is time for parents to teach young people early on that in diversity there is beauty and there is strength".

Maya Angelou, *Women of the Year Gala 2009*

As outlined in Chapter 2, enterprises in general and application landscapes in particular form complex systems. To survive in their environment, these systems need to balance exploration and exploitation activities with regard to their resources which are a major source for creating competitive advantage (see Chapter 2.3). When considering the complexity of application landscapes diversity plays a crucial role (see Chapters 2.4 and 3) although a detailed conceptual understanding in this context is still missing. Therefore, the goal of this chapter is to develop such conceptual understanding of diversity in the context of application landscapes. Based on EA meta-models and description logics, four different aspects of diversity are derived including **variation**, **variety**, **balance** and **disparity**. Thereby, this chapter provides the foundation for the upcoming diversity metric development in Chapter 5.

4.1. Description logics and enterprise architecture meta-models

To describe the architecture of an enterprise, the respective architect needs to be aware of relevant components as well as their relations. The components and relations to be modeled can, for example, be defined by an EA meta-model. Therefore, such meta-model defines the scope of an architectural description of an enterprise. To provide a holistic view on the enterprise, such meta-model typically contains both business related and information systems related components

as well as their interrelationships. Most EAM frameworks provide such EA meta-model, e.g. the TOGAF content metamodel (The Open Group, 2011), the ArchiMate language (Lankhorst, 2009) and different research approaches, for example the essential layers defined by Winter and Fischer (2007) or the information model patterns observed by Ernst (2008). Therefore, the actual meta-model used to develop architectural descriptions of an enterprise provides the basis for considering the diversity of modeled components and relations. It is important to note that EA meta-models are typically organization-specific because of individual information demands. Thus, no generally applicable meta-model satisfying all requirements currently exists.

To describe diversity of EA components, a more abstract modeling approach is required to be independent of concrete, organization-specific EA model and meta-model manifestations. In biology end ecology, for example, diversity of an ecosystem is described by the diversity of species living therein (MacArthur and MacArthur, 1961; Hill, 1973; Ying-Hui et al., 2014). In the context of economics, diversity of people and workforce has been considered based on ethnics (Alesina and La Ferrara, 2004) and gender (Campbell and Mínguez-Vera, 2008). As we can easily see, from all the different diversity approaches present in various disciplines, a more abstract model of systems under investigation can be derived. In each case the modeler needs to be aware of individuals (e.g. a person or an animal) and the concept which is used to uniformly describe a set of individuals (e.g. male or dog). Therefore, the formal approach of description logic seems to be appropriate to model a system under investigation, e.g. an application landscape, to allow consistent diversity assessments. To lay the foundation for the diversity framework developed in this chapter, description logics are introduced, followed by an application of description logic to EA meta-models.

4.1.1. Introduction to description logics

DLs are a family of formal knowledge representation languages which is used, for example, in artificial intelligence for formal reasoning on concepts of an application domain, e.g. in software design and analysis based on UML diagrams (see Berardi et al., 2005). Within DL intensional knowledge, i.e general knowledge about the problem domain, is represented by the **TBox**. Likewise, the extensional knowledge, i.e. knowledge which is specific to a particular problem, is represented by the **ABox**. Both can be defined as follows.

Definition: TBox
The TBox contains intensional knowledge in the form of a terminology and is built through declarations that describe general properties of concepts.

(Baader, 2003, p. 12)

Definition: ABox
The ABox contains extensional – also called assertional – knowledge that is specific to the individuals of the domain of discourse.

(Baader, 2003, p. 13)

In this context, intensional knowledge is usually considered to be stable whereas extensional knowledge is usually thought to be contingent and therefore subject to change. Therefore, it can be concluded that two fundamental components are sufficient to formally describe the knowledge of a problem domain, e.g., an application landscape: **concepts** and their relationships (TBox) and **individuals** assigned to concepts (ABox). The key relationship between concepts is the *is-a* relationship. For example, a woman can be defined by the following declaration:

$$Woman \equiv Person \sqcap Female \tag{4.1}$$

It is important to note that concepts have to be defined acyclic, i.e., that concepts are neither defined in terms of themselves nor in terms of other concepts that indirectly refer to them. In contrast, the ABox contains assertions about individuals usually called membership assertions. This is expressed by the following example provided by Baader (2003):

$$Female \sqcap Person(ANNA) \tag{4.2}$$

In this example, the individual ANNA is a female person. Given the above definition of woman, one can derive from this assertion that ANNA is also an instance of the concept *Woman*. Figure 4.1 illustrates the general relationship of individuals and concepts. Thereby, individuals are considered to be atomic elements which cannot be divided into smaller elements. Usually, they represent things existing in the real world which can be named. Concepts are used to group and thereby describe individuals which are similar with respect to a certain aspect. A single individual can be described by many concepts whereas a single concept can describe many individuals. In addition, concepts can form hierarchies, i.e. a concept can be defined by using other concepts. In the previous example, *Female* is a concept used to describe individuals directly whereas the concept *Women* is solely defined based on other concepts.

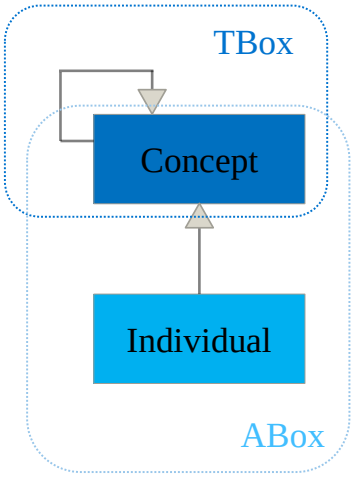


Figure 4.1.: General components of description logics

In literature, a variety of DLs can be found. These can be distinguished by the operators they allow. For example, Schmidt-Schauß and Smolka (1991) introduce the attributive language ($\mathcal{AL}$)

as a minimal language that is of practical interest. If A and B are used for atomic concepts, the letter R for atomic roles and the letters C and D for concept descriptions, the $\mathcal{AL}$ language provides the following syntax rules.

$$\begin{aligned} C, D \rightarrow A| & \text{ (atomic concept)} \\ \top| & \text{ (universal concept)} \\ \bot| & \text{ (bottom concept)} \\ \neg A| & \text{ (atomic negation)} \\ C \sqcap D| & \text{ (intersection)} \\ \forall R.C| & \text{ (value restriction)} \\ \exists R.\top & \text{ (limited existential quantification)} \end{aligned} \tag{4.3}$$

More complex versions can be generated by adding other constructors, e.g. union ($\mathcal{U}$), full existential quantification ($\mathcal{E}$), inverse properties ($\mathcal{I}$) and number restrictions ($\mathcal{N}$). Nevertheless, to describe the knowledge about EAs the $\mathcal{ALI}$ language is sufficient (see Chapter 4.1.3) which is created by adding inverse properties.

4.1.2. Examples of EA meta-models

The development of meta-models is a common approach to guide the description of knowledge about the architecture of an enterprise. These meta-models are similar to an ontology because they define the existing types of elements as well as their relationships. Instead of providing only general knowledge about description concepts, e.g. by description logics, such meta-models define not only what should be modeled but also what should not be modeled. Furthermore, they provide typical concepts used to model the architecture of an enterprise allowing direct application with minor adaptations instead of developing these concepts from scratch (e.g. a new TBox for EA for every company). To provide an overview about typical elements of an EA meta-model, exemplary approaches from literature and practice are reviewed in the following.

The previously mentioned industry standard TOGAF (The Open Group, 2011) contains the so called content metamodel which describes the concepts used in TOGAF. It is intended to provide guidance for organizations that wish to implement their architecture description within an EAM tool. Within the content metamodel, the following core terms are defined and correlated with each other:

Actor: A person, organization, or system that is outside the consideration of the architecture model, but interacts with it.

Application Component: An encapsulation of application functionality that is aligned to implementation structuring.

Business Service: Supports business capabilities through an explicitly defined interface and is explicitly governed by an organization.

Data Entity: An encapsulation of data that is recognized by a business domain expert as a discrete concept. Data entities can be tied to applications, repositories, and services and may be structured according to implementation considerations.

Function: Delivers business capabilities closely aligned to an organization, but not explicitly governed by the organization.

Information System Service: The automated elements of a business service. An information system service may deliver or support part or all of one or more business services.

Organization Unit: A self-contained unit of resources with goals, objectives, and measures. Organization units may include external parties and business partner organizations.

Platform Service: A technical capability required to provide enabling infrastructure that supports the delivery of applications.

Role: An actor assumes a role to perform a task.

Technology Component: An encapsulation of technology infrastructure that represents a class of technology products or a specific technology product.

In addition, the content metamodel provides six different extensions which can be used if required in a specific context. When all extensions are applied to the core content metamodel, a number of new metamodel entities are introduced. Figure 4.2 illustrates which entities are contained in the core content metamodel as well as which entities are introduced by extensions.

Another part of the TOGAF standard is the ArchiMate language (Lankhorst, 2009). It provides a more detailed meta-model and is supported by many architecture description tools, e.g. Archi[1] and BizzDesign architect[2]. Figure 4.3 illustrates the core concepts of the ArchiMate language. A detailed description of the elements is provided by the current ArchiMate specification[3].

In addition to industry standards like TOGAF and ArchiMate several research groups focused on the development of conceptual meta-models to guide the development of architectural descriptions of enterprises. For example, Winter and Fischer (2007) identified essential layers and artifacts of EA meta-models. These include:

- Strategy specification: hierarchy of organizational goals and success factors, product/service model (including partners in value networks), targeted market segments, core competencies, strategic projects, maybe business principles, dependencies between these artifacts
- Organization/process specification:
  - Specification of structure: organizational unit hierarchy, business location hierarchy, business role hierarchy (including skills requirements), dependencies between these artifacts
  - Specification of behavior: business function hierarchy, business process hierarchy in-

[1] http://www.archimatetool.com
[2] http://www.bizzdesign.com/tools/overview/
[3] http://pubs.opengroup.org/architecture/archimate2-doc/

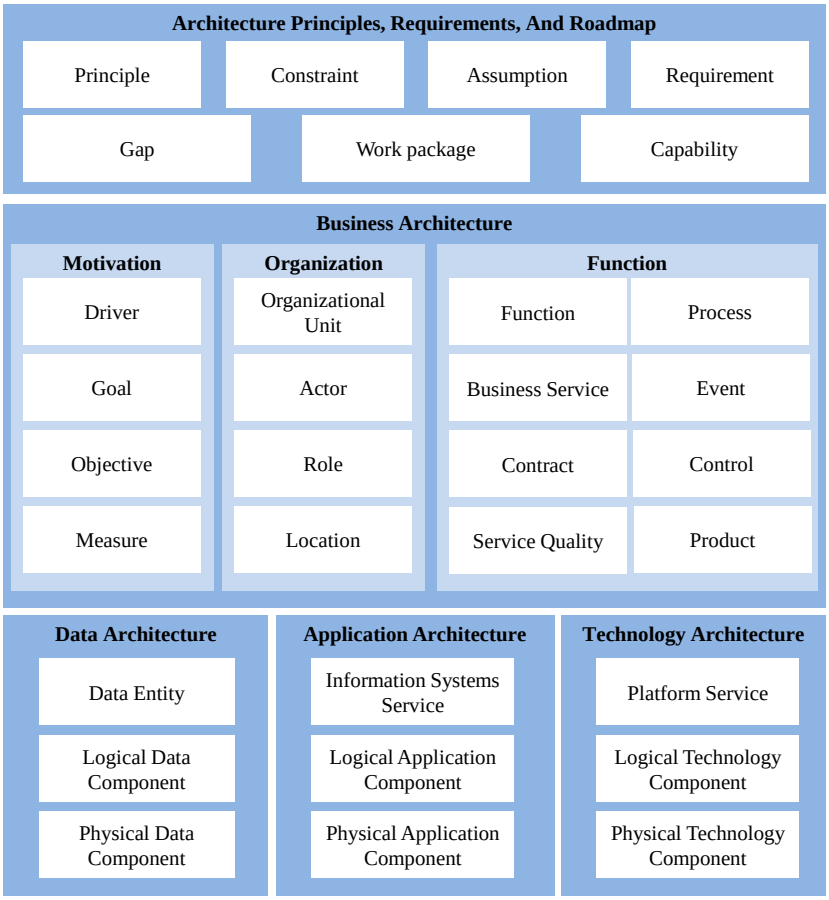


Figure 4.2.: TOGAF content metamodel with extensions according to The Open Group (2011)

cluding inputs/outputs (internal and external business services including service levels), metrics (performance indicators), service flows

  - Specification of information logistics: business information objects, aggregate information flows
  - Dependencies between these artifacts, e.g. responsibilities or information requirements

- Application specification:
  - Specification of applications and application components
  - Specification of enterprise services and service components
- Software specification:
  - Specification of software components: functionality hierarchy, message hierarchy
  - Specification of data resources: conceptual, logical and physical data models

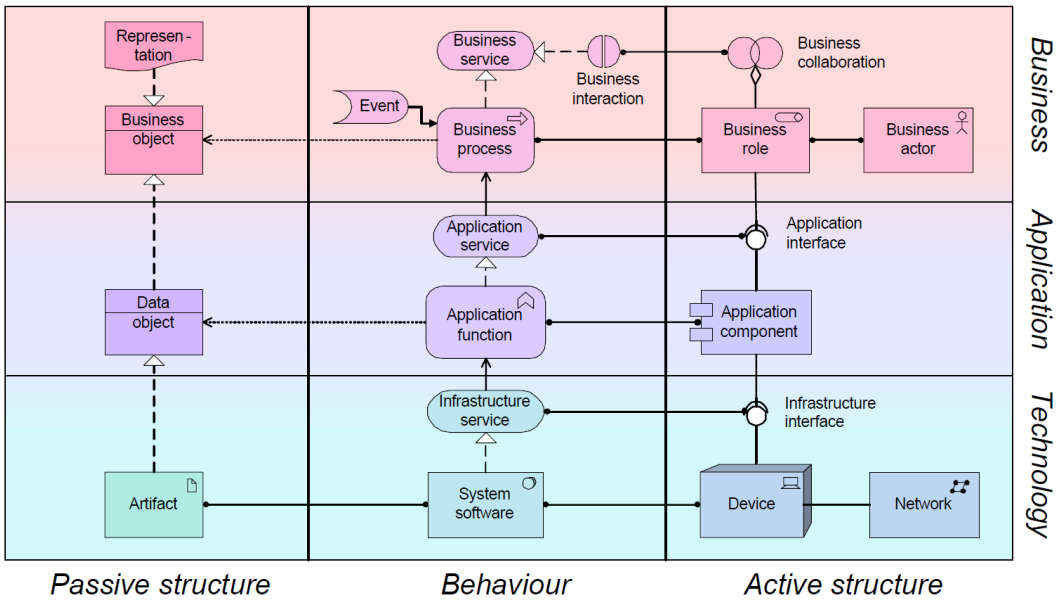


Figure 4.3.: Core concepts of the ArchiMate meta-model according to Lankhorst (2009, p. 91)

  - Dependencies between these artifacts, e.g. data usage by software components
- Technical infrastructure specification:
  - Specification of IT components: hardware units, network nodes, etc.
  - Dependencies between these artifacts
- Specification of dependencies between layers, e.g.:
  - Organizational goals/success factors vs. process metrics
  - Products/services vs. process deliverables
  - Organizational units vs. applications (ownership)
  - Activities vs. applications
  - Applications/enterprise services vs. conceptual data entity types
  - Applications/enterprise services vs. software components (composition)

In Ernst (2008), typical segments of EA meta-models observed in industry are collected. In this collection, some concrete patterns modeling more technical components can be found. For example, I-Pattern I-30 (Ernst, 2008, p. 199) includes a concept named *Support Relationship* which is used to model the relationship between *Business Applications*, *Organizational Units* and *Business Processes*. This concept is required, if the model should be able to express the fact that a *Business Application* is used within more than one *Business Process* by more than one *Organizational Unit*, but at least one of these *Organizational Units* uses another *Business Application* for at least one of these *Business Processes*.

Based on these exemplary descriptions of existing EA meta-models the following conclusions can be drawn:

1. An enterprise architecture description consists of individuals which are grouped by the concepts defined in the meta-model.
2. Relevant characteristics of individuals are described based on the attributes specified in the meta-model.
3. Concepts in the meta-model might be interrelated, e.g., within a hierarchy.
4. Concepts used within EA meta-models vary between frameworks and organizations.
5. Concepts presented in literature, frameworks and used in practice are often similar although addressed in different vocabularies, for example, business applications, business processes and infrastructure elements (see also Chapter 5.1 for a unified EA meta-model).

4.1.3. Diversity considerations based on different EA meta-model elements

As outlined by Baader (2003), the advantage of conceptual models, e.g., EA descriptions, is that they abstract away irrelevant details and allow more efficient examination of the current as well as past and projected future states of the universe of discourse, i.e., the enterprise. Likewise, DLs subscribe to an object-centered view of the world. Their ontology includes notions like individuals, their relationships and concepts grouping individuals into classes. Therefore, DLs provide the abstract formal definition of EA meta-model elements required to consider diversity of these elements without being tied to specific architecture frameworks. As outlined before, diversity is usually assessed on the concept level not on the level of individuals. By definition, individuals are separate entities which differ significantly. Thus diversity of two or more individuals can never be zero. This is different on the concept level. Depending on the concepts used to describe the individuals concepts can be defined to be equal, thus having no diversity. When applying the notions of individuals, roles and concepts to EA models in general and application landscapes in particular, a corresponding mapping is required.

First, some individuals might be quite concrete such as a person or a legal entity of a company. Others might be more abstract like a business goal or a business application. To be considered an individual, it is important to have an identity which allows them to be distinguished from each other and to be counted. What is to be considered an individual depends on what should be done with the information (Baader, 2003, p. 353). When modeling an application landscape, this imposes several questions which cannot be answered easily. For example, when considering an *infrastructure element* like an Oracle Database Server, each installation of this product can be regarded as an individual. This approach would be typical within the IT service management domain because each installation can be subject to failures independently of others. In the context of EAM, typically a different approach is preferred. The enterprise architect is usually only interested in the fact that Oracle Database Servers are part of the EA (see Chapter 5.3 for more examples derived from practice). The second question arising is how to model different versions of the same product. If the enterprise architect is interested in the existence of different versions of, for example, Oracle Database Servers, each version has to be modeled as an

individual. In such case, the Oracle Database Server itself cannot be an individual anymore but has to be modeled as concept.

Second, the concepts used to describe the set of relevant individuals forming the EA needs to be defined. This can be done, for example, by using or adapting one of the EA frameworks introduced in the previous chapter. Continuing with the aforementioned example, the concept Oracle Database Server can be introduced as an atomic/primitive concept to describe the different versions considered to be individuals. Furthermore, the concept Infrastructure Element can be introduced as another concept grouping all individuals being an Oracle Database Server as well as other individuals being, for example, DB2 Database Servers.

$$\begin{aligned} \text{Oracle Database Server} &\sqsubseteq \text{Infrastructure Element} \\ \text{DB2 Database Server} &\sqsubseteq \text{Infrastructure Element} \end{aligned} \tag{4.4}$$

Given such concept relation, diversity within a concept can be assessed by analyzing respective sub-concepts as well as their instances. This becomes particularly important if diversity should not only be analyzed for the whole application landscape but also for different areas within the application landscape, e.g. for different *domains* or *organizational units*.

Third, EA meta-models explicitly define relations between individuals and therefore concepts. For example, a typical relation is modeled between Business Applications and Infrastructure Elements. In such case, it is assumed that a Business Application is composed of several Infrastructure Elements. In DLs this can be expressed by roles. For example, the fact that business application A is composed of Oracle Database Server 10g would be described as follows:

$$\begin{aligned} &\text{Oracle Database Server(ORACLE DATABASE SERVER 10g)} \\ &\text{Business Application(A)} \\ &\text{composedOf(A, ORACLE DATABASE SERVER 10g)} \end{aligned} \tag{4.5}$$

Instead of stating that a Business Application is composed of Infrastructure elements one could also say that an Infrastructure Element is used by a Business Application. This could easily be expressed by defining an inverse role as follows:

$$\text{usedBy} \equiv (\textbf{inverse}\ \text{composedOf}) \tag{4.6}$$

This example demonstrates the necessity of the DL $\mathcal{ALI}$ to represent enterprise architecture knowledge because relationships are usually navigated in both directions.

Finally, it can be concluded that the application of DLs provides a formal way of representing EA knowledge contained in organization-specific EA models based on organization-specific or EA framework-specific meta-models. This enables the development of a comprehensive and organization-independent diversity framework using only the notions of individuals, roles and concepts, within the next chapters.

4.2. Variation: Differences between individuals

When considering diversity of application landscapes, at least four different dimensions have to be regarded (Stirling, 2007; Page, 2011). In this chapter, the first dimension, **variation**, will be introduced and illustrated with examples. Thereby, understanding variation will provide the basis for the other three dimensions because it distinguishes diversity among individuals from diversity among concepts.

4.2.1. Derivation and definition

In his comprehensive work on diversity and complexity, Page (2011) identified three fundamental types of diversity: (1) diversity within a type, (2) diversity of types and (3) diversity of compositions. Thereby, the notion of types is used to distinguish between diversity within types (variation) and across types (diversity). Although this approach should comply with the intuitive understanding of most people, if applied to a concrete domain, a formal definition of types and instances is required. As outlined before, this could be achieved, for example, by using Description Logics (DLs) to describe the universe of discourse. Thereby, an existing EA metamodel can be used. What Page (2011) calls a *type* is equivalent to a *concept* in DL. Likewise, *instances* represent the same notion as do *individuals*. Accordingly, the first dimension to be considered when assessing application landscape diversity can be defined as follows:

> **Definition: Variation**
> Variation is defined as diversity within a concept. This refers to differences in the amount of some attribute or characteristic.
>
> (Page, 2011, p. 20)

In general, "variation allows for local search, provides responsiveness to minor changes in the environment and serves as an engine for diversity of types" (Page, 2011, p. 3). It therefore plays an important role in enabling adaptability and robustness of complex systems. Considering, for example, birds belonging to the same species, the individuals exhibit variation in wing size and beak length. Not only can such differences produce a survivability advantage for some individuals of that species, they also allow the species to adapt to a changing environment. The demarcation of variation, i.e. differences among individuals, from diversity among concepts is crucial because it defines aspects which are not of interest at the concept level. Such approach is also typical in other disciplines, e.g., organizational science, where diversity is considered at unit level rather than considering focal unit member's differences from other members (Harrison and Klein, 2007). Figures 4.4 and 4.5 illustrate the meaning of variation both figuratively and exemplarily.

In Figure 4.4, the *variation* dimension of diversity, i.e., diversity among individuals, is illustrated by eight individuals which differ in terms of color and diameter. In this case, all individuals belong the same concept *shape*. Likewise, Figure 4.5 exemplifies the variation dimension of diversity by depicting five different animals which also differ in terms of their color. Note that the common concept used here is *animals* and not concrete species like *cats* and *dogs*. If the

Figure 4.4.: Figurative illustration of variation

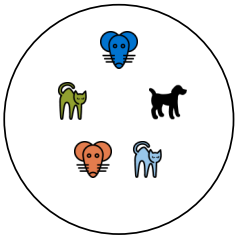


Figure 4.5.: Exemplary illustration of variation

concept used here would not be *animals* but *cats*, five different cats would have been depicted all differing in terms of their color and maybe size. Accordingly, what is considered to be variation depends on the modeling approach of a respective universe of discourse. Given such approach, variation lays the foundation for considering diversity among concepts.

4.2.2. Exemplary application

The notion of *variation* introduced in the previous chapter can easily be transferred to the context of application landscapes. Therefore, the individuals of interest need to be defined first. When considering, for example, Database Management System (DBMS), different individuals might be observed within an application landscape. Using DL, an exemplary application landscape might be modeled as follows.

Example 4.1: Variation within application landscapes: Databases. Here, five different DBMS individuals are used to exemplify variation:

$$\begin{aligned} &\textsc{Database} \sqsubseteq \textsc{Infrastructure Element} \\ &\textsc{Database}(\text{ORACLE } \# \ 371) \\ &\textsc{Database}(\text{ORACLE } \# \ 376) \\ &\textsc{Database}(\text{MongoDB } \# \ 21) \\ &\textsc{Database}(\text{MongoDB } \# \ 45) \\ &\textsc{Database}(\text{DB2 } \# \ 33) \end{aligned} \tag{4.7}$$

Given these five instances of DBMS, one can easily imagine that they may differ with respect to various characteristics, e.g., the number of tables in their schema, their vendor, the date of installation or the current amount of stored data. Nevertheless, all individuals belong to the same concept, i.e. Database. Therefore, differences between these individuals need to be distinguished from differences between concepts, e.g. between Databases and other Infrastructure Elements like operating systems.

As can be seen from the example above, it depends on the modeling approach at which level diversity is assessed. Here, concrete instances of database management systems have been regarded as individuals. Nevertheless, an enterprise architect might also consider a concrete product, e.g. ORACLE, as an individual depending on the questions to be answered. To ensure consistency, one of both approaches has to be chosen. In addition, variation within application landscapes cannot only be observed among infrastructure elements. For example, differences between business applications can also be regarded as variation. This is illustrated in the following example.

Example 4.2: Variation within application landscapes: Application. In this example, four different individuals, i.e. SAP business application instances, are used to exemplify variation:

$$\begin{aligned} &\text{SAP} \sqsubseteq \textsc{Business Application} \\ &\text{SAP(FI/CO \# 11)} \\ &\text{SAP(FI/CO \# 12)} \\ &\text{SAP(CRM \# 21)} \\ &\text{SAP(CRM \# 22)} \end{aligned} \tag{4.8}$$

In this example, two different instances of the "same" SAP application exist. Reasons include redundancy requirements or different application scopes like Europe and the US. Given these four instances of SAP applications one can easily imagine that these individuals differ with respect to various characteristics. Such characteristics include, but are not limited to: the number of users, the date of installation as well as the daily amount of transactions. Nevertheless, all individuals belong to the same concept, i.e. SAP. Therefore, the differences between these individuals need to be distinguished from differences between concepts, for example between SAP and other Business Applications like Microsoft applications.

As demonstrated by the two examples above, what is considered to be *variation* depends on the modeling approach of the enterprise architect. If the architect decided for a suitable approach, this diversity dimension provides the foundation for the remaining three diversity dimensions considering diversity among concepts (not among individuals), due to the fact that concepts used for grouping individuals always neglect a certain amount of diversity between them. By neglecting variation, the attention and respective analyses can focus on diversity aspects relevant to the enterprise architect.

4.3. Variety: Different concepts to describe a set of individuals

As outlined before, diversity can be distinguished into three types (Page, 2011). The previously described dimension of **variation** covers the first type, i.e. diversity among individuals. The second type, i.e. diversity among concepts, requires more detailed elaboration. A seminal work in this area has been presented by Stirling (2007). He distinguishes three different types of diversity among concepts: **variety**, **balance** and **disparity**. Because all of them are relevant to describe diversity in the context of application landscapes, all three aspects will be introduced in the following chapters.

4.3.1. Derivation and definition

"When people speak of diversity, they tend to mean differences of types" (Page, 2011, p. 22). Therefore, a clear distinction of diversity and variation is required. Based on achievements in other disciplines like ecology, economics, taxonomy, palaeontology, complexity and information theory, Stirling (2010) considers variety as necessary property of diversity. It is defined as follows.

> **Definition: Variety**
> Variety is the number of diverse categories of 'option' into which a system may be apportioned. It is the answer to the question: "how many options do we have?"
>
> (Stirling, 2010)

Variety is the fundamental aspect of diversity at the concept level. Differences between individuals (variation) are neglected. Only the number of concepts needed to describe the universe of discourse is of interest. Therefore, variety is similar to the category count already proposed by MacArthur (1965). To describe the diversity of ecosystems, counting the number of species is a fundamental approach. Other examples of variety considerations include the simple enumeration of firms or products in economics (Kauffman, 1993; Saviotti and Mani, 1995) or the counting of fuels or technologies in energy policy (UK Department of Energy, 1988). Therefore, it can be stated that *all else being equal, the greater the variety, the greater the diversity* (Stirling, 2007). Figures 4.6 and 4.7 illustrate the meaning of variety both figuratively and exemplarily.

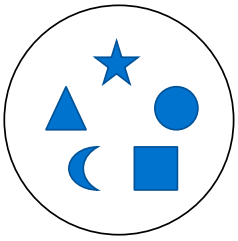


Figure 4.6.: Figurative illustration of variety

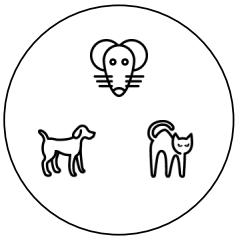


Figure 4.7.: Exemplary illustration of variety

Figure 4.6 depicts five different concepts to illustrate the *variety* dimension of diversity. In contrast to Figure 4.4 which shows eight individuals belonging to five different *shapes*, Figure 4.6

includes only one element for each of those concepts although a different amount of individuals per concept might exist within the universe of discourse. These individuals might as well be subject to variation that is not visible at the concept level. Given the example of species depicted in Figure 4.5, Figure 4.7 shows the corresponding variety by including one concept for each species, i.e. *dog*, *cat* and *mouse*, to describe individuals existing in the universe of discourse. Color and size are not represented because they form aspects of variation not variety.

4.3.2. Exemplary application

The notion of *variety* introduced in the previous chapter can easily be transferred to the context of application landscapes. Therefore, the concepts required to describe the individuals existing in the universe of discourse, i.e. the application landscape, need to be identified. As mentioned before, concepts considered relevant might vary from organization to organization. Usually, they can be derived directly from the EA meta-model in use. The following example describes how the variety of database management systems can be assessed if the EA meta-model contains a class named *Database* having an attribute *vendor*.

Example 4.3: Variety within application landscapes: Databases. Based on the *vendor* of a *database* (which is an *infrastructure element*) concepts can be derived which are able to classify instances of database management systems used by the company under investigation. As pointed out by Baader (2003) modeling the two required conditions for each concept separately is essential in this context.

$$
\begin{aligned}
\textsc{Database} &\sqsubseteq \textsc{Infrastructure Element} \\
\textsc{Database} &\sqsubset (\textbf{the}\ \text{vendor}\ (\textbf{one-of}\ \texttt{'Oracle}\ \texttt{'IBM}\ \texttt{'MongoDB})) \\
\textsc{Oracle DB} &\sqsubseteq \textsc{Database} \\
\textsc{Oracle DB} &\sqsubset (\textbf{fills}\ \text{vendor}\ \texttt{'Oracle}) \\
\text{IBM DB} &\sqsubseteq \textsc{Database} \\
\text{IBM DB} &\sqsubset (\textbf{fills}\ \text{vendor}\ \texttt{'IBM}) \\
\textsc{MongoDB} &\sqsubseteq \textsc{Database} \\
\textsc{MongoDB} &\sqsubset (\textbf{fills}\ \text{vendor}\ \texttt{'MongoDB})
\end{aligned}
\tag{4.9}
$$

By counting the required concepts to describe the individuals in this example the variety of the universe of discourse can be assessed. Here, three different concepts are used which are by definition disjoint. Note that variation, as presented in Example 4.1, is not considered here.

As can be seen in the example above, it depends on the modeling approach how diversity of database management systems is assessed. Here, concepts have been derived based on the vendor

of each individual. Other approaches might use the product line or the paradigm of each database management system individual. In addition, variety within application landscapes can not only be observed among infrastructure elements. For example, the amount of required concepts to describe business applications can also be regarded. This is illustrated in the following example by introducing concepts for applications which are compliant to the target landscape and for those which are not.

Example 4.4: Variety within application landscapes: Application. In this example, four different individuals, i.e. SAP business application instances which have been used earlier, are used to exemplify variety. They are distinguished based on their compliance to the target architecture. In this example, SAP FI/CO individuals are considered to be compliant with the target architecture but SAP CRM individuals are not considered to be compliant because other tools, e.g. Salesforce[4], should be used in the future.

$$
\begin{aligned}
\textsc{Business Application} &\sqsubset (\textbf{the}\ \text{compliant}\ (\textbf{one-of}\ \texttt{'yes}\ \texttt{'no})) \\
\textsc{Compliant BAs} &\sqsubseteq \textsc{Business Application} \\
\textsc{Compliant BAs} &\sqsubset (\textbf{fills}\ \text{compliant}\ \texttt{'yes}) \\
\textsc{Non-Compliant BAs} &\sqsubseteq \textsc{Business Application} \\
\textsc{Non-Compliant BAs} &\sqsubset (\textbf{fills}\ \text{compliant}\ \texttt{'no})
\end{aligned}
\tag{4.10}
$$

In this example, two disjoint concepts are used to describe the individuals of business applications with regard to their compliance to the target architecture. Given these concepts one can easily assess the variety of business applications which is two. Compared with the previous example of database management systems the variety in this example is lower because fewer concepts have been used. Again, differences among individuals belonging to the same concept are neglected because this aspect belongs to the variation dimension.

As demonstrated by the two examples above, variety can be clearly distinguished from variation especially if description logics are used to model the universe of discourse, i.e the application landscape. Again, variety assessments also depend on the modeling approach used by the enterprise architect but typical concepts can be derived from existing EA meta-models. This offers flexibility to assess diversity in general and variety in particular from different viewpoints. Existing complexity and diversity approaches already regard variety of application landscapes although addressed in different vocabularies (Mocker, 2009; Schuetz et al., 2013; Schneider et al., 2015c).

[4] http://www.salesforce.com

4.4. Balance: Distribution of individuals among concepts

The second type of diversity among concepts identified by Stirling (2007) is **balance**. This dimension will be introduced in this chapter and applied to the context of application landscapes by providing two examples.

4.4.1. Derivation and definition

In addition to variety, which regards only the number of existing concepts, balance also regards the distribution of instances among these concepts. It can be defined as follows.

> **Definition: Balance**
> Balance is a function of the pattern of apportionment of individuals across concepts. It is the answer to the question: "how much of each type of thing do we have?"
>
> (Stirling, 2007)

Similar to variety, balance abstracts from differences between individuals (variation) but reflects the fact that a different amount of individuals per concept might be present. In ecology, balance is often also referred to as *evenness* (Pielou, 1969) whereas *concentration* is often used in economics (Finkelstein and Friedberg, 1967). Therefore, different measures of balance have been developed, including, for example, Shannon entropy (Shannon, 1948), Gini index (Gini, 1912) and Herfindahl-Hirschman index which is used by American authorities to calculate market concentrations (Shapiro, 2010). In the context of EA, balance has also been referred to as *heterogeneity* (Schuetz et al., 2013). Independently of a concrete measure, balance and therefore diversity increases the more equal the distribution of individuals among the existing concepts is. The more dominant a single concept is with respect to its number of individuals the lower the overall balance is, thus diversity decreases. It can be concluded, that *all else being equal, the more even is the balance, the greater the diversity* (Stirling, 2007). Figures 4.8 and 4.9 illustrate the meaning of balance both figuratively and exemplarily.

Figure 4.8.: Figurative illustration of balance

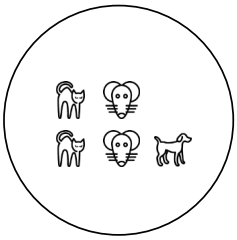


Figure 4.9.: Exemplary illustration of balance

Figure 4.8 depicts five different bars representing the amount of individuals belonging to the respective concept to illustrate the balance dimension of diversity. The bars are ordered from left to right in decreasing order. Although not necessary, this visualization facilitates the interpretation of balance within the universe of discourse. In this dimension, individuals are considered

again but differently from the variation dimension. As outlined in Chapter 4.2, variation regards the differences between individuals. In contrast, balance treats all individuals belonging to the same concept equally but distinguishes concepts based on their relative frequency. For the example used previously, Figure 4.9 illustrates balance with the three concepts cat, dog and mouse having a different amount of individuals, i.e., one or two.

4.4.2. Exemplary application

The notion of balance introduced in the previous chapter can easily be transferred to the context of application landscapes. Therefore, the individuals existing in the universe of discourse, i.e. the application landscape, have to be assigned to their respective concepts. Again, the concepts used to describe these individuals might be company-specific. The following examples demonstrate different balances for two hypothetical application landscapes.

Example 4.5: High balance within application landscapes. In this example, the concepts of Example 4.3 are used combined with the individuals already used in Example 4.1. Here, five different individuals belong to three different concepts which are all subsets of the Database concept. In addition, another individual belonging to the IBM DB concept has been added.

$$
\begin{aligned}
&\textsc{Database} \sqsubseteq \textsc{Infrastructure Element}\\
&\textsc{Database} \sqsubset (\textbf{the}\ \text{vendor}\ (\textbf{one-of}\ \text{'Oracle 'IBM 'MongoDB}))\\
&\textsc{Oracle DB} \sqsubseteq \textsc{Database}\\
&\textsc{Oracle DB} \sqsubset (\textbf{fills}\ \text{vendor}\ \texttt{'Oracle})\\
&\text{IBM DB} \sqsubseteq \textsc{Database}\\
&\text{IBM DB} \sqsubset (\textbf{fills}\ \text{vendor}\ \texttt{'IBM})\\
&\textsc{MongoDB} \sqsubseteq \textsc{Database}\\
&\textsc{MongoDB} \sqsubset (\textbf{fills}\ \text{vendor}\ \texttt{'MongoDB})\\
&\textsc{Oracle DB}(\text{ORACLE \# 371})\\
&\textsc{Oracle DB}(\text{ORACLE \# 376})\\
&\textsc{MongoDB}(\text{MongoDB \# 21})\\
&\textsc{MongoDB}(\text{MongoDB \# 45})\\
&\text{IBM DB}(\text{DB2 \# 33})\\
&\text{IBM DB}(\text{DB2 \# 30})
\end{aligned}
\tag{4.11}
$$

In this example, the individuals are equally distributed among three different concepts. Therefore, this application landscape exhibits high balance and thus high diversity. Note that variation is not considered here.

In contrast, the following example shows an application landscape exhibiting low balance among database management systems due to concentration of instances within one concept.

Example 4.6: Low balance within application landscapes. In this example, the concepts of the previous example are used but different individuals are assumed.

$$
\begin{aligned}
&\textsc{Database} \sqsubseteq \textsc{Infrastructure Element}\\
&\textsc{Database} \sqsubset (\textbf{the}\ \text{vendor}\ (\textbf{one-of}\ \texttt{'Oracle}\ \texttt{'IBM}\ \texttt{'MongoDB}))\\
&\textsc{Oracle DB} \sqsubseteq \textsc{Database}\\
&\textsc{Oracle DB} \sqsubset (\textbf{fills}\ \text{vendor}\ \texttt{'Oracle})\\
&\textsc{IBM DB} \sqsubseteq \textsc{Database}\\
&\textsc{IBM DB} \sqsubset (\textbf{fills}\ \text{vendor}\ \texttt{'IBM})\\
&\textsc{MongoDB} \sqsubseteq \textsc{Database}\\
&\textsc{MongoDB} \sqsubset (\textbf{fills}\ \text{vendor}\ \texttt{'MongoDB})\\
&\textsc{Oracle DB}(\text{ORACLE \# 371})\\
&\textsc{Oracle DB}(\text{ORACLE \# 376})\\
&\textsc{Oracle DB}(\text{ORACLE \# 377})\\
&\textsc{Oracle DB}(\text{ORACLE \# 379})\\
&\textsc{MongoDB}(\text{MongoDB \# 45})\\
&\textsc{IBM DB}(\text{DB2 \# 30})
\end{aligned}
\tag{4.12}
$$

Here, the individuals are clearly concentrated within the Oracle DB concept. Therefore, this application landscape exhibits low balance and thus low diversity.

As demonstrated by the two examples above, balance can be determined based on the amount of individuals assigned to each concept used to describe an application landscape. Therefore, it is interrelated with the variety dimension because the more concepts are used, the higher the balance becomes. Note that if only a single concept is used, balance will always be zero. In addition, balance treats all concepts equally important thus assuming a nominal scale. Balance does not answer the question whether the desired concepts have more individuals than non-desired concepts. Only the distribution of individuals is considered. Existing complexity and diversity approaches already regard balance (e.g. Schuetz et al., 2013) although referred to as heterogeneity.

4.5. Disparity: Degree of difference among concepts

The third dimension of diversity among concepts identified by Stirling (2007) is **disparity**. It considers the degree of difference among used concepts and is explained in this chapter.

4.5.1. Derivation and definition

In addition to variety and balance, Stirling (2007) derived a third dimension of diversity among concepts: **disparity**. It can be defined as follows.

> **Definition: Disparity**
> Disparity refers to the manner and degree in which the concepts may be distinguished. It is the answer to the question: "how different from each other are the concepts of thing that we have?"
>
> (Stirling, 2007)

Considering disparity is essential because the balance dimension and all its metrics do not consider the degree to which the analyzed concepts differ (Page, 2011). This implies, for example, that without considering disparity a basket with three apples, three pears and three oranges exceeds the diversity of another basket with seven apples, one orchid and one pencil. In both baskets the number of concepts (variety) is equal but the distribution of individuals among these concepts (balance) is more concentrated in the second basket; thus the second basket exhibits less diversity. Intuitively, most people would assign higher diversity to the second basket because of larger differences of things included. Therefore, the two former dimensions need to be complemented by the disparity dimension to cover all relevant aspects of diversity. In fact, it is judgments over disparity, which necessarily govern the resolving of categories used to characterize variety and balance (Stirling, 2007). Figures 4.10 and 4.11 illustrate the meaning of disparity both figuratively and exemplarily.

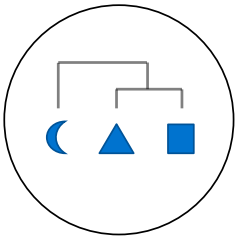


Figure 4.10.: Figurative illustration of disparity

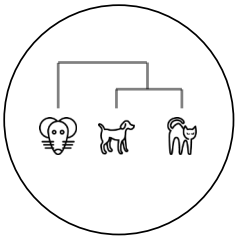


Figure 4.11.: Exemplary illustration of disparity

Figure 4.10 depicts three of the aforementioned concepts. Their degree of difference, i.e. disparity, is indicated by a dendrogram. A dendrogram is often used to illustrate the arrangement of clusters produced by hierarchical clustering (Everitt, 1998, p. 96). In this case, square and triangle are considered to be more similar to each other than both are compared to the moon.

The reason is that square and triangle are both made of straight lines whereas the moon is constructed by two curved lines. A similar example is shown in Figure 4.11 to indicate the disparity of species. Here, dogs and cats are considered to be more similar than they are compared to mice. The foundation for creating the dendrogram is the taxonomy used in ecology and biology to classify species based on the time back to the most recent common ancestor (Weitzman, 1992). Dogs and cats both belong to the group of carnivore and their most recent common ancestor lived about 55 million years ago (Solé et al., 2014). In contrast, the most recent common ancestor of mice and both cats and dogs belongs to the group of boreoeutheria which is about 100 million years old (O'Leary et al., 2013). In general, it can be concluded that *all else being equal, the more disparate are the represented elements, the greater the diversity* (Stirling, 2007). Because disparity has not yet been considered in the context of application landscape management (see Chapter 5.2), two examples are provided in the next chapter.

4.5.2. Exemplary application

Applied to the context of application landscapes, disparity can be observed among various concepts. To get sensible results, disparity should be considered among sub-concepts. The following example demonstrates disparity among database management systems—a sub-concept of infrastructure elements.

Example 4.7: High disparity within application landscapes. In this example, the concepts of the previous example are used to describe an application landscape exhibiting high disparity.

$$
\begin{aligned}
&\textsc{Database} \sqsubseteq \textsc{Infrastructure Element}\\
&\textsc{Database} \sqsubset (\textbf{the}\ \text{vendor}\ (\textbf{one-of}\ \texttt{'Oracle 'IBM 'MongoDB}))\\
&\textsc{Oracle DB} \sqsubseteq \textsc{Database}\\
&\textsc{Oracle DB} \sqsubset (\textbf{fills}\ \text{vendor}\ \texttt{'Oracle})\\
&\textsc{IBM DB} \sqsubseteq \textsc{Database}\\
&\textsc{IBM DB} \sqsubset (\textbf{fills}\ \text{vendor}\ \texttt{'IBM})\\
&\textsc{MongoDB} \sqsubseteq \textsc{Database}\\
&\textsc{MongoDB} \sqsubset (\textbf{fills}\ \text{vendor}\ \texttt{'MongoDB})
\end{aligned}
\tag{4.13}
$$

Given these concepts, disparity can be considered to be high, because the concept MongoDB differs in many aspects from the other to concepts. For example, it does not follow the relational paradigm and is not supported by a very large vendor. If the concept MongoDB would be replaced by a concept SQL Server, the disparity would decrease because concepts would then be more similar.

Similar to infrastructure elements, disparity can also be considered for business applications. The following example demonstrates how low disparity can be observed, for example, based on technical characteristics of business applications.

Example 4.8: Low disparity within application landscapes. In this example, two different concepts are used to describe the individuals of the concept Business Application in the area of customer relationship management which exhibit low disparity.

$$\begin{aligned} \textsc{Business Application} &\sqsubset (\textbf{the}\ \text{vendor}\ (\textbf{one-of}\ \texttt{'SAP}\ \texttt{'Oracle})) \\ \textsc{SAP CRM} &\sqsubseteq \textsc{Business Application} \\ \textsc{SAP CRM} &\sqsubset (\textbf{fills}\ \text{vendor}\ \texttt{'SAP}) \\ \textsc{Siebel CRM} &\sqsubseteq \textsc{Database} \\ \textsc{Siebel CRM} &\sqsubset (\textbf{fills}\ \text{vendor}\ \texttt{'Oracle}) \end{aligned} \tag{4.14}$$

Given these concepts, disparity can be considered to be low, because the two concepts are very similar with regard to the defined aspects. For example, the individuals of both concepts are all on-premise solutions. If the concept Siebel CRM would be replaced by a concept Salesforce, the disparity would increase because concepts would then be less similar since Salesforce is a cloud-based solution.

The examples above illustrate that the actual distribution of individuals among the used concepts is not considered in the disparity dimension. Nevertheless, disparity assessments depend on the (subjective) selection of criteria used to describe the difference between two or more concepts. Such characteristic define the distance between concepts. The larger this distance gets, the greater the diversity.

4.6. A unified view on application landscape diversity

In previous chapters different dimensions of diversity have been introduced. In this chapter, these dimensions are unified to a comprehensive framework which is sufficient to describe and assess diversity of application landscapes. Thereby, interrelations between these dimensions are revealed as well as limits of their manageability.

4.6.1. Conceptual framework

Based on the different dimensions of diversity introduced in the previous chapters, a unified framework for application landscape diversity can be derived. Thereby, the framework needs to

distinguish between diversity among individuals (*variation*) as well as diversity among concepts used to describe these individuals. The latter includes three different dimensions, i.e. *variety*, *balance* and *disparity*. If applied to a specific universe of discourse, i.e. an application landscape, the distinction between individuals and concepts is fundamental. Therefore, variation is placed in the middle of the framework which is depicted in Figure 4.12.

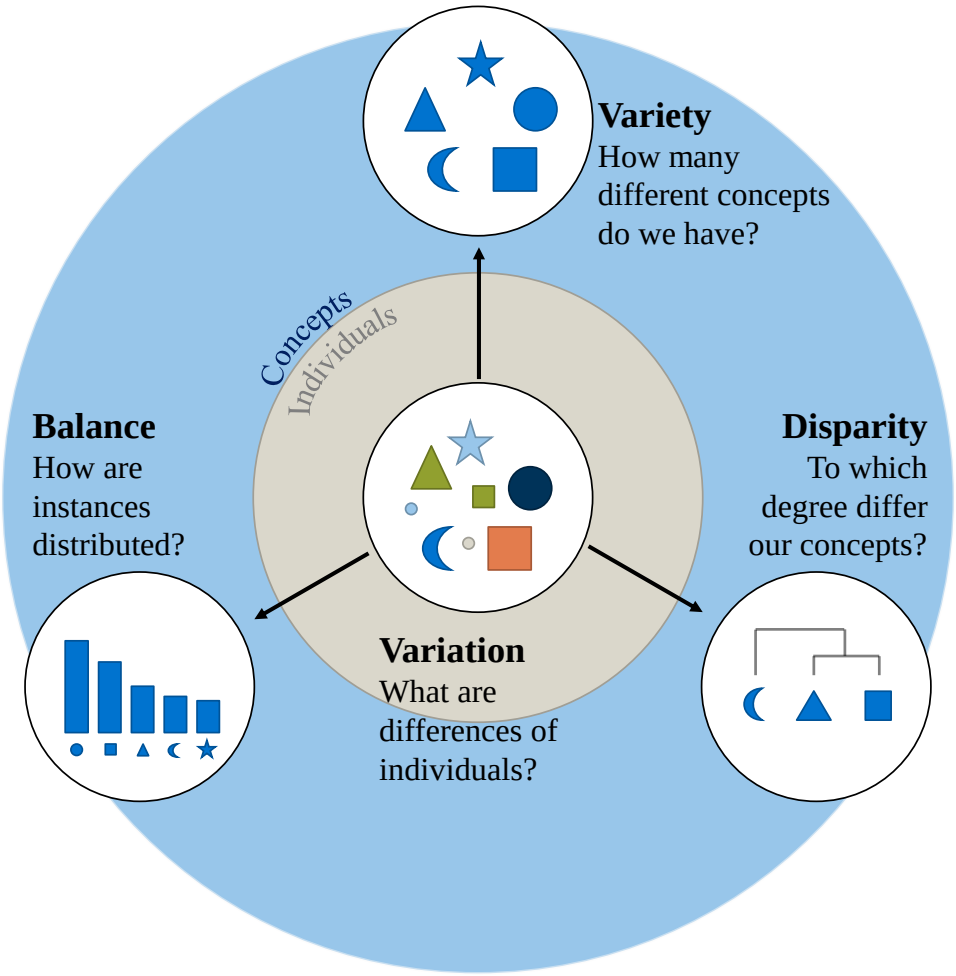


Figure 4.12.: Unified diversity framework for application landscapes following Stirling (2010)

Based on the demarcation of variation in a specific universe of discourse, the concepts required to describe instances can be derived, e.g. from an EA meta-model. When considering diversity one moves from the gray area in the middle of the diversity framework to the blue area including these concepts. There, three different dimensions need to be distinguished. First, *variety* regards the number of different concepts used to describe individuals. The more concepts are used the greater the diversity. Second, *balance* regards the pattern of distribution of individuals among these concepts. The more similar the distribution is to an equal distribution the greater the diversity. Third, *disparity* regards the degree to which concepts in use differ from each other. The greater the distance between these concepts the greater the disparity and therefore the diversity of the universe of discourse. Despite multiple disciplines and various contexts, “there

seems no other obvious candidate for a fourth important general property of diversity beyond these three" (Stirling, 1998). The examples given in the previous chapters demonstrate how each of these three dimensions can be observed within application landscapes.

Although all three dimensions are necessary to describe diversity each one by its own is insufficient. Nevertheless, each dimension constitutes the other two (Stirling, 2007). This in turn highlights difficulties with existing diversity concepts as well as associated indicators that focus exclusively on subsets of these dimensions (Eldredge, 1992). If management decisions would be based on such limited understandings of diversity the resulting application landscape might not exhibit the intended properties. Regarding *variety* and *balance*, for example, requires a previous understanding of *disparity*. As May (1990) points out, an ecological community comprising 20 varieties of beetle is less diverse than the one comprising less than 20 species drawn from different insect, reptile and mammalian taxa. Likewise, an electricity system is less diverse if it comprises equal contributions from brown coal, lignite, oil and gas than if it is an equal mix of coal, nuclear and renewable energy (Stirling, 1994). However, a category like *renewable energy* might itself be considered to be highly diverse if it is equally apportioned into wind, solar, hydro, tidal, biomass, landfill gas, etc. The focus of attention in each case is neither on variety nor balance, but on disparity (Stirling, 2007). In the context of application landscapes, taking variety or balance as proxies for diversity can thus be highly sensitive to subjective construction and partitioning of taxonomies. For example, if diversity of DBMSs should be assessed, the diversity might be large due to several concepts and an equal distribution (e.g. SQL and NoSQL databases as well as databases specialized for OLAP and for OLTP). But the set of relational databases which forms a subset of the database concept might exhibit low diversity.

The need for considering all three dimensions together is demonstrated by the following example. Figure 4.13 shows three different concepts (variety) used to describe database management system individuals of two different application landscapes. In addition, a dendrogram indicates the assumed distance between these concepts (disparity) complemented by bars indicating the relative frequency of each concept with regard to the number of its individuals (balance).

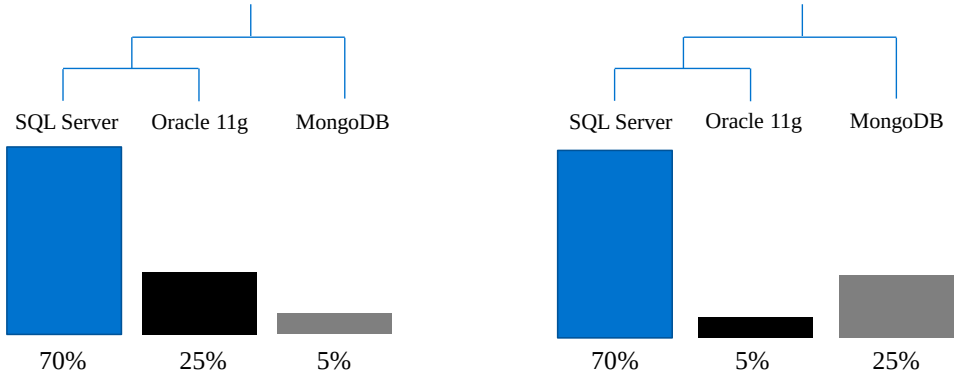


Figure 4.13.: Exemplifying the need for an integrated view of three diversity dimensions

Note that the only difference between these application landscapes is that the relative frequencies of concepts Oracle and MongoDB have been changed. When all three dimensions of diversity are considered independently, the diversity of both application landscapes would be considered

to be equal. In both cases three different concepts are used (equal diversity). Because the same concepts are used the degree of difference between these concepts is equal (equal disparity). By assessing the relative frequencies of each concept, the degree of concentration is equal in both cases because the relative frequency values are equal (equal balance). Nevertheless, by considering all three dimensions together, one would attribute higher diversity to the second, i.e. right, application landscape, due to fact that the relative frequency of the more exotic concept MongoDB is higher compared to the other case. In addition, given the deliberations above, it is also obvious that a change in variety is likely to cause a change in balance as well as in disparity. This needs to be considered especially when applying metrics to measure diversity.

4.6.2. Applications to enterprise architectures

As outlined earlier, the diversity framework described in the previous chapter can be applied to arbitrary EA descriptions. To support decisions regarding an increase or decease of diversity within an application landscape, describing diversity is not sufficient. Instead, the specifics of an EA, especially dependencies between its elements, need to be considered during diversity management. Typically, enterprise architectures are divided into several layers (see Chapter 4.1.3). Therefore, changing the diversity within the application landscape might not be possible in the general case due to dependencies to other layers or cross-cutting aspects. Therefore, diversity within the application landscape needs to be distinguished in *inter-layer induced diversity* as well as *intra-layer induced diversity* which are defined as follows.

Definition: Inter-layer induced diversity
Inter-layer induced diversity covers the amount of diversity with respect to variety, balance and disparity which is induced by individuals or components belonging to another EA layer.

Definition: Intra-layer induced diversity
Intra-layer induced diversity covers the amount of diversity with respect to variety, balance and disparity which is solely induced by individuals or components belonging to the same EA layer.

The following example illustrates the need for such distinction.

Example 4.9: Inter-layer induced diversity. Imagine a business process P being part of the business architecture requires the use of a specific business application A. Imagine further, that another, similar business process Q requires the use of another business application B due to its design. Neither business application A is able to support business process Q nor business application B is able to support business process P.

In such case, the diversity appearing on the application landscape layer, i.e. two different concepts with some kind of individual distribution having a significant disparity, is induced by individuals belonging to another layer.

Example 4.10: Intra-layer induced diversity. Imagine a business process being part of the business architecture requires the use of a customer relationship management application. Imagine further, that another process requires also the use of a customer relationship management application. Therefore, two different customer relationship management applications are in use.

If two different applications are used to satisfy the requirements mentioned in Example 4.10, this diversity is considered as intra-layer induced. Both business processes consider the customer relationship management application as a black box. The same would be true if, for example, two different business applications require a relational database but two different products are used to satisfy this requirement.

It is important to note that changing, e.g. reducing, intra-layer induced diversity might be easier because changes have to be made only locally. In the case of inter-layer induced diversity changing diversity might be more complicated or time-consuming because dependent individuals belonging to other EA layers might have to be adapted as well. Figure 4.14 visualizes the differences between inter-layer and intra-layer induced diversity by showing explicit dependencies among layers causing inter-layer induced diversity and omitting more general dependencies (e.g. requirements for a relational database) allowing for intra-layer induced diversity.

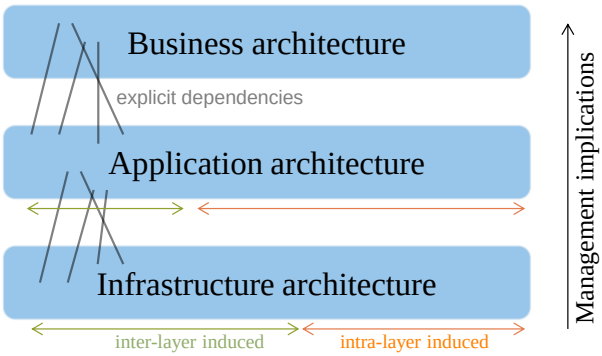


Figure 4.14.: Demarcation of inter- from intra-layer induced diversity

CHAPTER 5

Measuring structural complexity and diversity of application landscapes

"You can't control what you can't measure."

Tom DeMarco, *Controlling Software Projects*

In this chapter, indicators are developed to quantify diversity of application landscapes—one aspect contributing to an application landscape's complexity. These indicators provide decision support for enterprise architects by enabling them to monitor the evolution of their application landscape's diversity and measure the impact of standardization or diversification activities. After describing the research approach in detail, indicators already used in practice are assessed (see Chapter 5.1). Identified indicator patterns are then complemented by indicators identified by a thorough literature review (see Chapter 5.2). Together with several companies from Germany and Switzerland all indicators suitable to quantify complexity in general and diversity in particular are assessed regarding their utility for practitioners (see Chapter 5.3). To allow a full quantification of all aspects contributing to an application landscape's diversity—as presented in Chapter 4—two new indicators are proposed to complement existing indicators (see Chapters 5.4.3 and 5.4.4). Finally, a prototypical implementation of a software solution supporting the calculation of complexity and diversity indicators is presented (see Chapter 5.5).

To lay the foundation for this chapter it needs to be noted that in literature the term *metric* is not always defined in a similar manner. For example, Fenton and Pfleeger (1997, p. 323) consider a metric to be "any number extracted from a software entity" and state that each measure is a metric but not vice versa. In contrast, in mathematics a metric is defined as a "measure of distance" (Poels and Dedene, 2000). Any function $\delta : X \times X \rightarrow R$ is a "measure of distance" if

it satisfies the following conditions:

$$\begin{aligned} \text{non-negativity: } & \forall x, y \in X : \delta(x, y) \geq 0 \\ \text{identity: } & \forall x, y \in X : \delta(x, y) = 0 \Leftrightarrow x = y \\ \text{symmetry: } & \forall x, y \in X : \delta(x, y) = \delta(y, x) \\ \text{triangle inequality: } & \forall x, y, z \in X : \delta(x, y) \leq \delta(x, z) + \delta(z, y) \end{aligned} \tag{5.1}$$

Accordingly, for this work a metric will be defined as follows:

Definition: Metric
A metric is a measure of distance which satisfies the conditions of *non-negativity, identity, symmetry* and *triangle inequality.*

In this chapter, the term measure or indicator is used to denote numbers derived from software entities as long as the requirements of a metric are not fulfilled.

5.1. Complexity and diversity indicators used in practice

The first step towards the development of application landscape diversity indicators is to identify and analyze solutions already developed and implemented in practice. By following the Pattern-based Design Research method (Buckl et al., 2013) these solutions should be documented by using the EAM pattern language (Buckl et al., 2007). The EAM pattern language distinguishes three different kinds of patterns:

M-Patterns Methodologies define steps to be taken in order to address given concerns. Furthermore, as a guidance for applying the method, statements about its intended usage context are provided, which include the concerns to which the methodology can be applied. Sometimes these patterns are also called Process Patterns (Moser et al., 2009).

V-Patterns Viewpoints provide the languages used by methodologies. A viewpoint proposes a way to present data stored according to one or more information model patterns.

I-Patterns Information models represent underlying models for the data visualized in one or more viewpoints. An information model pattern conveys an information model fragment including the definitions and descriptions of the used information objects.

In addition to these three types of EAM patterns and in accordance with the IEEE/ISO standard 420101 (International Organization for Standardization, 2007), the EAM pattern language includes a list of recurring concerns. They serve as an entry point and help to select appropriate patterns within a given context based on stakeholder's requirements. Figure 5.1 depicts the conceptual model underlying the EAM pattern language. Note that a problem can be divided into a goal and a concern and only the concern is relevant for EAM patterns. In addition, Schneider and Matthes (2015) proposed to include metric patterns as a special kind of visualization pattern.

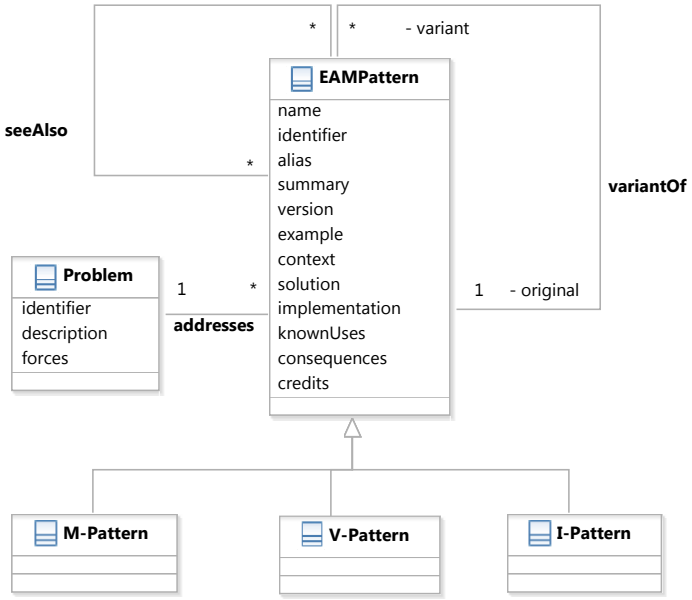


Figure 5.1.: Conceptual model of the EAM pattern language (Ernst, 2008)

5.1.1. Data collection process

In November 2013, six different organizations and one consulting company have been asked to share their currently implemented approaches to manage and measure application landscape complexity and diversity. These companies have been selected due to their experience in and maturity of their EAM in general and complexity management in particular. Every participant has been involved in complexity management activities at his/her organization for more than one year. As shown in Table 5.1, most organizations are active in the financial service sector on a global scale and are headquatered either in Germany or Switzerland. They employ between 8,000 and more than 100,000 Full Time Equivalents (FTEs). As indicated by the last column, the application landscapes to be managed vary between 157 and more than 5,000 business applications. Every organization delegated one or two enterprise architects to present their current approach to complexity and diversity management in a one day workshop. Based on these presentations and subsequent queries pattern candidates could be identified and documented. Afterwards, the technique of hermeneutics (see Gadamer, 1975) has been used to develop a deeper understanding of these indicators as well as their context.

5.1.2. Observed metric patterns

The data collection described in the previous chapter resulted in the observation of varying approaches to measure and manage application landscape diversity. The smallest approach consists of only four indicators whereas the most extensive approach consists of 22 indicators with specialized software support for their calculation and visualization. By conducting the hermeneutic circle and having conversations with the enterprise architects in order to remove

Id	Industry sector	Headquarter	Employees	Number of applications
1	Banking	Switzerland	>10,000 FTEs	157
2	Banking	Germany	>8,000 FTEs	247
3	Banking	Germany	>52,000 FTEs	1,898
4	Insurance	Germany	>14,000 FTEs	234
5	Banking	Switzerland	>46,000 FTEs	3,734
6	Automotive	Germany	>100,000 FTEs	>5,000

Table 5.1.: Companies involved in metric pattern collection

ambiguity, six indicators which were used by at least one half of the participating companies could be identified (see also Schneider et al., 2015c). These indicators are based on the following sets of EA entities and relationships:

$$
\begin{aligned}
A &\text{ denotes the set of all business applications} \\
D &\text{ denotes the set of all functional domains} \\
A_{buy} &\text{ denotes the set of all COTS business applications} \\
A_{buy} &\subseteq A \\
A_{make} &\text{ denotes the set of all custom business applications} \\
A_{make} &\subseteq A \\
A_{buyCustomize} &\text{ denotes the set of all customized COTS business applications} \\
A_{buyCustomize} &\subseteq A \\
A_{buy} \cap A_{make} \cap A_{buyCustomize} &= \emptyset \\
I &\text{ denotes the set of all infrastructure components} \\
F &\text{ denotes the set of all business function} \\
assigned &\subseteq A \times D \\
connected &\subseteq A \times A \\
uses &\subseteq A \times I \\
functionPoints &\subseteq A \times \mathbb{N} \\
supports &\subseteq A \times F
\end{aligned}
\tag{5.2}
$$

Baesd on this formal and consistent terminology the observed metric patterns can be described as follows:

Number of Applications: 4 out of 6 companies count the total number of business applications they have in their application landscape as well as the number of business applications belonging to a specific domain (d) given a relation called *assigend* which maps applications

to domains (see Equation 5.4). A higher number is associated with a higher application landscape or domain complexity.

$$Applications := |A| \tag{5.3}$$

$$Applications_d := |\{a \mid a \in A \wedge d \in D \wedge assigned(a, d)\}| \tag{5.4}$$

Number of Information Flows: 6 out of 6 companies count the number of interfaces or information flows each business application has based on a relation named *connected*. A higher number is associated with a higher application complexity. Thereby, an information flow is usually considered as a directed relationship between two business applications. Therefore, the *connected* relation is not necessarily symmetric.

$$Informationflows := |\{(a, b) \mid a, b \in A \wedge connected(a, b)\}| \tag{5.5}$$

Customization Level: 4 out of 6 companies assess the customization level of their applications. Therefore, they classify their applications as *buy*, *make* or *buy and customize* resulting in three disjoint subsets of A. The more business applications belong to the *buy* category, i.e. $|A_{buy}|$, the least complex is the application landscape. This metric can also be calculated for a single domain (see Equation 5.7). The maximum value of this metric indicating a complex application landscape or domain which consists only of customized business applications ($A_{buyCustomize} = A$) is 5, given the definition of the *customization* function shown in Equation 5.6. Respectively, the minimum value indicating a lest complex application landscape or domain is 1 ($A_{buy} = A$).

$$\begin{aligned} CustomizationLevel &:= \frac{\sum_{a \in A} customization(a)}{|A|} \\ customization(a) &= \begin{cases} 1 \text{ if } a \in A_{buy} \\ 3 \text{ if } a \in A_{make} \\ 5 \text{ if } a \in A_{buyCustomize} \end{cases} \end{aligned} \tag{5.6}$$

$$\begin{aligned} CustomizationLevel_d &:= \frac{\sum_{a \in A_d} customization(a)}{|A_d|} \\ A_d &:= |\{a \mid a \in A \wedge d \in D \wedge assigned(a, d)\}| \end{aligned} \tag{5.7}$$

Number of Infrastructure Elements: 4 out of 6 companies count the number of infrastructure components used to realize a business application according to Equation 5.8. A higher number is associated with a higher application complexity.

$$InfrastructureElements_a := |\{i \mid i \in I \wedge uses(a, i)\}| \tag{5.8}$$

Functional Scope: 3 out of 6 companies assess the functional scope for each business application. According to Equation 5.9, it is determined either by the application's function points or by the number of business functions realized by the particular application. A higher functional scope is associated with a higher complexity.

$$\begin{aligned} FunctionalScope_1 &:= \sum_{a \in A} functionPoints(a) \\ FunctionalScope_2 &:= |\{f \mid f \in F \wedge a \in A \wedge supports(a, f)\}| \end{aligned} \tag{5.9}$$

Functional Redundancy: 6 out of 6 companies assess if different applications provide the same functionality. A higher rate of functional redundancy is associated with a higher complexity since single changes then affect multiple applications. Therefore, this metric is closely related to diversity management.

$$Redundancy_f := |\{a \mid a \in A \wedge f \in F \wedge supports(a, f)\}| \tag{5.10}$$

A complete documentation of these metric patterns according to the structure required by the EAM pattern language (Matthes et al., 2012; Schneider and Matthes, 2015) can be found in Appendix A.

5.1.3. Integrated information model

To establish a common understanding of the information necessary to calculate the identified metric patterns described above an information model for each metric pattern has been derived by consistently modeling the required sets of EA elements and relations among them as introduced in Equation 5.2. The model elements used here adhere to the definitions used by the EAM Pattern Catalog (see Buckl et al., 2008) for compatibility reasons and the actual fit with practitioners' understanding. For example, metric pattern *Number of Applications* is defined on the set of all Business Applications whose elements are linked to a Functional Domain (see Equation 5.4). The metric is then to be calculated by summing up the number of Business Applications which belong to a particular Functional Domain. Metric pattern *Number of Information Flows* is defined on the same set of Business Applications and adds a concept named Information Flow which links two Business Applications (one as source and the other as target of the information flow). Although this could be modeled by an association in the Unified Modeling Language (UML), it is modeled as distinct class because practitioners usually add more information to describe an Information Flow, e.g. the protocol it uses. The calculation of metric pattern *Customization Level* is based on a property of the concept Business Application which is called Customization Level. This property can assume values like *buy*, *make* or *buy and customize* which can be modeled by an enumeration type. To calculate the *Number of Infrastructure Elements* an additional concept Infrastructure Component needs to be included. The calculation of diversity metric pattern *Functional Scope* requires either the concept of Function Points or the concept of Business Functions. Therefore, Business Applications can either have a property Function Points or they can be linked to one or

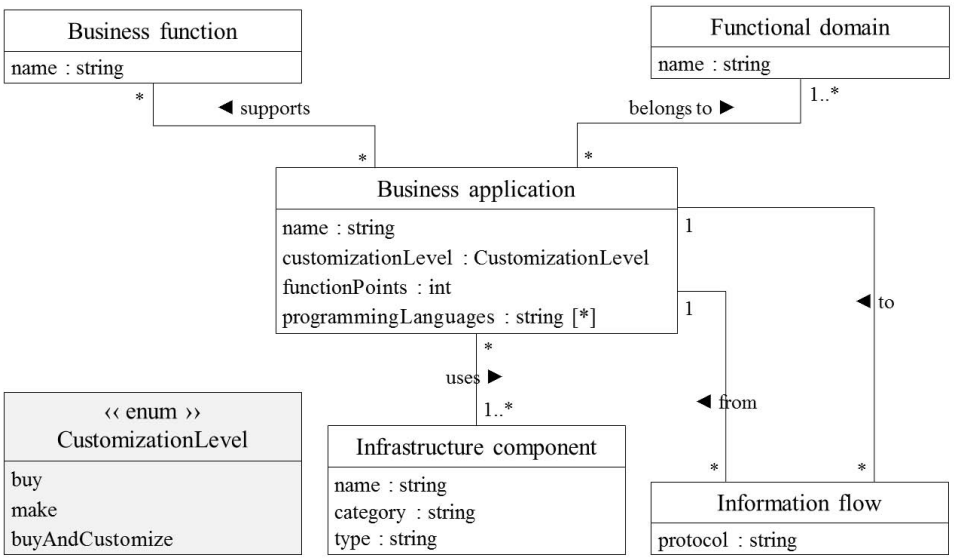


Figure 5.2.: An information model representing the data needed to calculate observed diversity metric patterns (Schneider et al., 2015c)

more Business Functions to explicitly model which functionality is provided instead of just modeling the amount of functionality. The calculation of diversity metric pattern *Functional Redundancy* requires the inclusion of the concept Business Function which is linked to all Business Applications supporting it. If one particular Business Function is supported by more than one Business Application the functional redundancy increases.

By integrating all concepts necessary to calculate the identified metric patterns an integrated information model can be derived. Such model is visualized in Figure 5.2. It can be used to define the information which has to be collected when diversity metric patterns should be implemented or which metric patterns can already be implemented based on available information.

5.2. Complexity and diversity indicators presented in literature

In this chapter, complexity and diversity indicators found in literature are presented. First, the data collection process, i.e. a literature review, is outlined followed by a detailed description of its results.

5.2.1. Data collection process

To completely reconstruct the accumulated knowledge in the domain of complexity and diversity indicators which forms the basis for the subsequent indicator development a literature review is necessary to complement metric patterns observed in practice (vom Brocke et al., 2009). This literature review is realized according to the guidelines presented by Webster and Watson (2002) and follows the process suggested by vom Brocke et al. (2009). Defining the review scope is the

Characteristic	Categories			
Focus	Research outcomes	Research methods	Theories	Applications
Goal	Integration	Criticism	Central issues	
Organization	Historical	Conceptual	Methodological	
Perspective	Neutral representation	Espousal of position		
Audience	Specialized scholars	General scholars	Practitioners	General public
Coverage	Exhaustive	Exhaustive & selective	Representative	Pivotal

Figure 5.3.: Instantiated taxonomy of literature reviews according to Cooper (1988)

first step. vom Brocke et al. (2009) suggest to use the framework provided by Cooper (1988) which is visualized and instantiated by highlighting relevant characteristics in grey in Figure 5.3. In order to identify existing indicators used for complexity or diversity quantification in the context of application landscapes the conducted literature review focuses on *research outcomes* with the goal of identifying *central issues*. The results are *organized conceptually*, *presented neutrally* and targeted at an audience of *scholars* in general. The results presented here include only a sample that typifies larger groups of articles to be *representative*.

5.2.2. Identified quantification approaches

The next step is to conceptualize the topic by reviewing seminal textbooks or handbooks first (Baker, 2000) and performing a subsequent concept mapping (Rowley and Slack, 2004). Therefore, the books by Saha (2014), Goetze and Jenssen-Waud (2013) and Page (2011) have been analyzed. During the subsequent concept mapping process the insights already achieved during the development of the conceptual basis (see Chapter 2.4.1) have been used additionally to identify key concepts as well as respective synonyms. The query "enterprise architect*" AND (diversity OR variety OR heterogeneity OR standard*) AND (metric OR measure) exemplifies the final search strings which has been used to consult the following scientific databases: EBSCOhost, Science Direct, ISI Web of Knowledge (Web of Science database), ACM, IEEE, and AIS. During the search period (10/22/2014 to 10/31/2014) 93 publications have been identified in total. After screening their titles and abstracts 84 of them had to be removed because they do not treat the concept of diversity as a central subject matter. As suggested by Webster and Watson (2002), for the nine remaining articles a forward and backward search has been performed. Accordingly, Table 5.2 summarizes representative articles about complexity and diversity metrics applicable in the context of application landscapes.

Boh and Yellin (2007) analyze to what extent the use of EA standards helps organizations to improve sharing and integrating IT resources across the enterprise. Therefore, the authors quantify the heterogeneity of infrastructure components and physical IT by counting the number of different technologies in use. Consequently, the approach only considers the *variety* aspect of diversity. *Balance* and *disparity* aspects are disregarded. The authors conclude that the use of EA standards has a significant effect on reducing diversity of IT infrastructure components.

	Standardization	Variety	Balance	Disparity	Complexity
Boh and Yellin (2007)	✓	✓			✓
Keuntje and Barkow (2010)	✓	✓			✓
Lagerström et al. (2014)					✓
Mocker (2009)	✓	✓		✓	✓
Schuetz et al. (2013)	✓	✓	✓		✓

Table 5.2.: Representative results of the diversity metric identification literature review

Keuntje and Barkow (2010) argue that standardization is a considerable activity to manage application landscape complexity. Thereby, standardization is understood as an activity which initially constitutes some particular technologies to be standard technologies and then tries to minimize the number of technologies which are not standard technologies. To assess the current degree of standardization the authors propose a metric named conformity index (CI). It is defined as the number of standard conforming technologies divided by the total number of technologies in use:

$$CI := \frac{amount\_of\_standard\_technologies}{total\_amount\_of\_technologies} \tag{5.11}$$

As one can see in Equation 5.11, when measuring the amount of diversity within an application landscape, the authors only consider the actual number of concepts/types. Therefore, their diversity notion is limited to the *variety* aspect while *balance* and *disparity* aspects are disregarded.

Lagerström et al. (2014) identify areas of high complexity within application landscapes by applying an approach known from software engineering to reveal the hidden external structure between software applications. Based on data about information flows between software applications the Design Structure Matrix approach is applied (see Steward, 1981). If the corresponding algorithm classifies the architecture of an application landscape as core-periphery architecture, the core applications are considered to be more complex than periphery applications. This assumption is based on the fact that, given the cyclic dependencies within the core, changes to those applications might propagate through the application landscape and cause additional and unforeseen costs. In addition to this classification the authors propose an indicator called propagation cost (PG) which is based on the VFI_i (Visibility Fan-In) of each software application (N) defined as the number of software applications that directly or indirectly depend on this particular software application. The PG indicator is defined as follows:

$$PC := \frac{\sum_{i=1}^{N} VFI_i}{N^2}, i \in N \tag{5.12}$$

Consequently, the authors consider application landscape complexity without referring to standardization in general or any diversity aspect in particular.

Mocker (2009) analyzes the causes and impacts of application landscape complexity. Therefore, he derived four different constructs in order to assess the complexity of a given application landscape: interdependency, diversity of technologies, deviation from technology standards and overlap/redundancy. These constructs are measured by specific indicators. Interdependency (I) is thereby defined as the sum of all incoming and outgoing interfaces of all software applications (N).

$$I := \sum_{i=1}^{N} interfaces\_out + interfaces\_in, i \in N \tag{5.13}$$

The diversity of technologies ($Diversity_{tech}$) is defined as the number of different types used as operating system or database management system. Therefore, T can either be set of all operating system products or database management products.

$$Diversity_{tech} := \sum_{i=1}^{T} i, i \in T \tag{5.14}$$

The deviation from technology standards construct is operationalized by one indicator for operating systems and one for database management systems. Both are calculated based on following exemplary definition:

$$Deviation := 1 - \frac{number\_of\_standard\_compliant\_operating\_systems}{number\_of\_operating\_systems} \tag{5.15}$$

The overlap/redundancy construct is measured by the overlap indicator which is defined as the portion of supported business processes which are also supported by another software application. Therefore, it describes the degree to which an application covers certain functionality already covered by other applications:

$$Overlap := 1 - \frac{number\_of\_supported\_processes\_covered\_by\_others}{number\_of\_supported\_processes} \tag{5.16}$$

Analyzing these indicator definitions (Equations 5.13 – 5.16), one can conclude that Mocker (2009) regards only the variety aspect of diversity, e.g. in Equation 5.14, and a particular disparity aspect, e.g. in Equation 5.16. Hence, he neglects the balance aspect of diversity. The author refrains from providing general applicable metric definitions.

Schuetz et al. (2013) define a general measure of EA complexity based on the amount and heterogeneity of both EA elements and their relationships. To calculate heterogeneity the authors analyze various measures of heterogeneity and conclude that Shannon entropy is best suitable for EA heterogeneity quantification. Therefore, the entropy measure (EM) is defined as:

$$EM := -\sum_{i=1}^{n} p_i ln(p_i) \; with \; p_i := \frac{x_i}{\sum_{j=1}^{n} x_j} \tag{5.17}$$
$$x_i := absolute\ number\ of\ elements\ assigned\ to\ characteristic\ i$$

By using Shannon's entropy, Schuetz et al. (2013) account for both *variety* and *balance* when measuring diversity within application landscapes. Hence they neglect the *disparity* aspect of diversity.

All articles identified by this literature review are concerned with the complexity of application landscapes. With the exception of Lagerström et al. (2014) all other authors propose standardization as means to reduce complexity. Therefore, several indicators are proposed to calculate application landscape diversity in order to monitor standardization success. As can be seen in Table 5.2, variety is the common aspect of diversity which is considered by all those indicators. Furthermore, Schuetz et al. (2013) provide the only approach which generally accounts for the balance aspect of diversity. The aspect of disparity is generally unaccounted for. Only Mocker (2009) uses a very basic notion of it to define his indicator for the functional redundancy of business applications. Therefore, we can conclude that a comprehensive approach to diversity quantification which accounts for all three aspects (variety, balance, disparity) is missing. Thereby, a generally applicable metric to calculate disparity for arbitrary EA model elements has not been developed yet. Furthermore, it remains unclear which of the identified metrics is actually useful for practitioners.

5.3. Benefits and drawbacks of existing indicators

The analysis of existing indicators to quantify application landscape diversity described in the previous chapters revealed that scholars propose several metrics while practitioners recurrently use others. Therefore, the question of interest is which of those indicators are actually useful for practicing enterprise architects. Schneider et al. (2015c) state that there are basically two different possibilities to evaluate the suitability of the identified complexity and diversity indicators: quantitatively and qualitatively. A quantitative evaluation could, e.g., additionally measure consequences of high application landscape complexity or diversity and then calculate the correlation coefficient for both indicators. In case of a strong correlation the respective metrics could be considered useful due to their predictive ability. Nevertheless, there are some issues associated with such evaluation approach. First, the cause-and-effect relationships between complexity/diversity and other variables, e.g. costs, might be non-linear, delayed in time or simply unknown. Furthermore, the cause-and-effect relations might be multi-causal instead of mono-causal meaning that there might be other causes having the same effect as well as other effects having the same cause. Therefore, a qualitative evaluation approach is preferred here.

The first step towards a qualitative evaluation of identified indicators' utility is to define utility in this context. In general, utility can be defined as the ability of something to satisfy needs or wants. According to the economic utility theory of Neumann and Morgenstern (1953) utility reflects the preferences of decision makers by assigning a number to each decision maker's alternatives to convey their relative attractiveness. Therefore, it is required to identify characteristics of complexity and diversity indicators which determine their utility for enterprise architects. Because preferences of organizations and individual enterprise architects might vary and an absolute scale for numerical utility measurement is absent it is refrained from measuring total utility in this context which is in line with Pareto (1906). In stead, the results of an assessment considering relevant characteristics will be provided which allows enterprise architects to make better informed decisions about implementing one or more of the identified complexity and diversity indicators.

5.3.1. Assessment framework

To develop an assessment framework for the utility of complexity and diversity indicators applicable for application landscapes, enterprise architects from six different organizations as well as one EA consultant have been asked for their assessment criteria during a half-day workshop in 2013. After several group discussions consensus about the following relevant characteristics has been reached:

Management suitability: Complexity and diversity indicators for application landscapes are used to inform decisions of higher level managers. Therefore, such indicators need to be understandable without greater effort. Understandability in this context includes but is not limited to simple calculation instructions, comprehensible impact analyses, summable metric results and intuitive visualizations. A metric can thus be suitable either for general managers or just for specialists.

Data collection effort: All indicators rely on appropriate EA descriptions. The creation of such descriptions requires a significant amount of data collection efforts. Therefore, the necessary data collection effort should be kept to a minimum for complexity and diversity indicators. Accordingly, required data can either be typically available, collectable with additional effort or require a large data collection.

Capability to express perceived complexity: Usually, enterprise architects already have an individual idea about which parts of their application landscapes show high complexity or diversity. In general, complexity and diversity indicators should be able to formally express this subjective instinct to a sufficient degree. Therefore, an indicator can either comply to the perceived complexity and diversity of enterprise architects or not.

Clarity: A complexity or diversity indicator provides clarity if it is defined in an unambiguous fashion. In this context unambiguous means that the calculation description does not give rise to any discussions about its validity. Thus an indicator definition can be regarded to be either arbitrary, ambiguous or clear.

Figure 5.4 visualizes the four different characteristics which will be used to assess the utility of identified complexity and diversity indicators. In addition, it provides concrete manifestations for each characteristic.

Characteristic	Manifestation		
Management suitability	Suitable for specialists	Suitable for general managers	
Data collection effort	Data typically available	Additional effort	Large effort
Perception compliance	Compliant	Not compliant	
Clarity	Arbitrary	Ambiguous	Clear

Figure 5.4.: Framework for complexity and diversity indicator utility assessment

5.3.2. Evaluation data description

To apply the evaluation framework introduced in the previous chapter and to assess the utility of the complexity and diversity indicators identified by the literature review those indicators need to be applied to real-world data. Therefore, four different companies have been asked to share their EA descriptions and allow for respective calculations. All four companies are part of the financial service sector and have already participated in the industry metric analysis described in Chapter 5.1. For the evaluation companies with ids 1, 2, 3 and 4 shared their EA data (see Table 5.1). To ensure comparability of indicator results across different companies the organization-specific data models had to be transformed to conform with the integrated information model visualized in Figure 5.2. These transformations include the renaming of classes and attributes, reverting the direction of relations and the transformation of attributes to relations, and vice versa. Thereby, it was possible to apply a logical mapping from the existing individual information model concepts to the corresponding core model concepts in each of the four cases.

As outlined by Schneider et al. (2015c), the four companies provided data according to Table 5.3. Therein, an overview of consolidated data is provided by giving the number of entities for every type and respectively the ratio of maintained values for attributes and relations of the corresponding type. A hyphen means that the corresponding company does not maintain the corresponding concept and thus could not provide it. The relations uses (Operating system) and uses (Database system) of type Business application are referring to specific categories of Infrastructure components. These are the only specific categories of Infrastructure components which are provided by at least two of the four companies. An Infrastructure component's type refers to its vendor or component class, e.g., Windows could be the type of Windows XP. When asked about the data quality the enterprise architects could not promise complete data but all of them are satisfied with the data quality and use this data for decision support in their organizations.

5.3.3. Indicator assessment results

In order to assess the utility of all identified complexity and diversity indicators the first step is to apply them to real-world data. This has been done in four different cases as described in the previous chapter (see Table 5.3). The extent to which companies are able to provide the required data will be used to assess the characteristic *data collection effort*. Then, based on the calculation results and corresponding interviews with respective enterprise architects the *compliance with their perception* is assessed. Both characteristics *management suitability* and *clarity* have then been assessed during a half-day workshop with all participating enterprise architects. Please note that the metrics presented by Mocker (2009) are not evaluated separately because they are mostly included in the set of indicators observed in practice. The only indicator not covered is the *overlap* indicator for which no evaluation data could be obtained. In the following the results of the evaluation are presented.

Concept	C1	C2	C3	C4
Business application	157	247	1898	234
- customization level	100 %	89 %	100 %	100 %
- function points	-	99 %	24 %	98 %
- programming languages	-	45 %	-	72 %
- implements (Use Case)	-	-	-	-
- supports (Business function)	-	97 %	-	-
- belongs to (Functional domain)	100 %	99 %	87 %	100 %
- uses (Operating system)	-	67 %	90 %	83 %
- uses (Database system)	-	48 %	60 %	67 %
Business function	-	580	-	-
Functional domain	13	16	33	41
Infrastructure component (Operating system)	-	9	15	12
- type	-	-	100 %	100 %
Infrastructure component (Database system)	-	10	8	10
- type	-	-	-	-
Information flow	539	827	8252	1214
- protocol	100 %	-	100 %	100 %
- from (Business application)	100 %	100 %	100 %	100 %
- to (Business application)	100 %	100 %	100 %	100 %

Table 5.3.: Indicator evaluation data (Schneider et al., 2015c)

5.3.3.1. Observed industry indicators

In Chapter 5.1 several complexity and diversity indicators have been observed recurrently in practice. To assess the utility of these indicators according to the assessment framework presented in Chapter 5.3.1 the goal is to calculate all these indicators in every of the previously presented cases. Due to the fact that not every company was able to provide the necessary data not every indicator could be calculated in every case. Accordingly, the actually calculated indicators are summarized in Table 5.4. Therein, for every indicator the ranges as well as the average value are included. For example, the *Number of Information flows* per application provides insights about an application landscape's connectedness. Although in Case 3 an application with a maximum of 319 information flows could be observed—which is almost three times as much information flows as the most connected application in Case 4 (122) has—the average number of information flows is much smaller in Case 3 compared to Case 4 (8.7 instead of 10.4). In addition, the *Customization level* indicator shows how different the analyzed application landscapes are regarding their respective sourcing strategy. For example, in Case 2 the range values reveal that there exists a domain which includes only customized standard software (5 is the maximum value), which is mostly an unwanted situation according to the respective enterprise architect. Likewise, in Case 3 one can see that there exists a domain which consists only of unmodified standard software products. Nevertheless, the comparability of some indicators is limited due to different underlying models. For example, the minimum and maximum values of the *Functional scope* indicator for Cases 3 and 4 highlight this fact. Furthermore, the indicator results need to

Indicator	C1	C2	C3	C4
Application-level metrics				
Number of Information flows	[0;108], 2.5	[0;70], 6.7	[0;319], 8.7	[0;122], 10.4
Number of Infrastructure components	-	[0;20], 4.7	[0;22), 0.4	[0;13], 3.9
Function scope (Function points)	-	[1;34], 6.7	[1;56], 7.3	[10;400], 178.9
Functional scope (Business functions)	-	[0;34], 6.7	-	-
Domain-level metrics				
Number of Applications	[6;28], 14	[1;53], 16	[1;194], 50	[6;88], 25
Number of Information flows	[10;195], 83	[1;390], 110	[0;1316], 491	[23;580], 282
Customization level	[1.4;3.9], 3	[1.4;5], 2.7	[1;3.5], 2.5	[2;3.3], 2.5
Number of Infrastructure components	-	[0;103]; 7.6	[0;44], 11.5	[9;34], 19.5
Functional scope (Function points)	-	[0;559], 22	[0;295], 67	[440;13810]; 4655
Functional scope (Business functions)	-	[0;543], 22	-	-

[min;max], average

Table 5.4.: Industry metric calculation results (Schneider et al., 2015c)

be analyzed on a more fine-grained (domain-specific) level to reveal more insights. It is unlikely, for example, that a company has a general sourcing strategy for all domains. Instead, different strategies are used for different domains and therefore the indicator analysis needs to be done in more detail than is possible here (Schneider et al., 2015c). Based on the actual indicator calculation, subsequent interviews with enterprise architects and a group discussion among all participating enterprise architects, it can be summarized that the complexity and diversity indicators identified in industry are suitable for general managers, do not require additional data collection efforts because most of the data is usually available, are compliant with experts' perceptions of complexity and diversity but some are ambiguous regarding their definition due to the lack of a coherent theory. These assessment results are visualized in Figure 5.5.

5.3.3.2. Heterogeneity indicator evaluation

The heterogeneity-based diversity indicator proposed by Schuetz et al. (2013) is defined in a generic fashion. Therefore, it has to be instantiated before it can be applied to the EA data provided by four different companies. Because there exist no guidelines for deriving such concrete

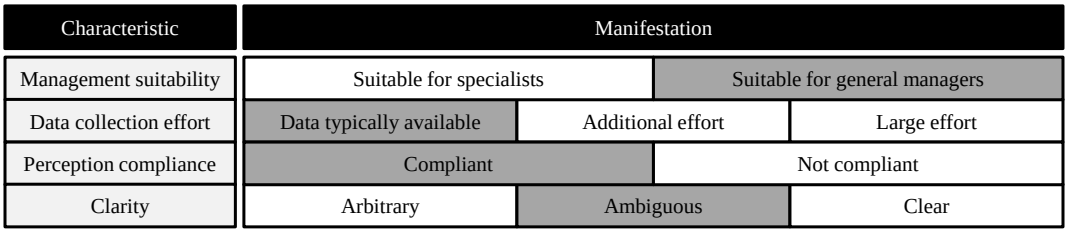


Figure 5.5.: Observed industry indicator assessment results

instantiations the given data is used as a foundation and all sensible possibilities are analyzed. Table 5.5 provides a detailed overview about all derived heterogeneity indicators. Due to the scope of this thesis and with regard to the available data all indicators are either defined on the application landscape layer (A) or the infrastructure layer (I). The analyzed heterogeneity metric is applicable for both EA entities (T) and relationships (R) which is also indicated in Table 5.5. Instead of providing all calculation results for all indicators in every case—which is of limited interest here—the indicator *Coupled domain* heterogeneity will be highlighted because it provided interesting insights into the architectures of the companies and because this indicator was realizable in all four cases. First, it will be explained more precisely how this indicator was calculated and what insights enterprise architects can derive from this indicator. As can be seen from Table 5.5, the indicator *Coupled domains* focuses on the application layer and architecture element's relations. The indicator considers the interfaces of applications assigned to a functional domain D and the heterogeneity of the functional domains, a given domain D is coupled with (Schneider et al., 2015c). Coupling between domains is determined by the information flows of the applications assigned to these domains.

Example 5.1: Coupled domain heterogeneity. Let application APP1 be assigned to domain D1 and let it have three interfaces to three other applications, each assigned to a distinct domain. In this case, the interfaces of APP1 would be equally distributed among three domains. Now imagine an application APP2 assigned to domain D1 having three interfaces to three applications, all assigned to domain D2. Because of the concentration of the interfaces on one domain, the functional heterogeneity of the interfaces of APP2 is lower than the functional heterogeneity of the interfaces of APP1.

To compare calculated values of *Coupled domain* heterogeneity with target values such target values have to be defined first. During the aforementioned workshop such target values have been developed. According to the enterprise architects, applications assigned to a value generating domain, e.g. loans, trades or sales in case of banking institutions, should—in principle—have low functional heterogeneity. Usually, those applications should share their data and services only with a limited amount of other systems, e.g. a data warehouse, outside of their domain.

Indicator	A	I	T	R	Cases	Interpretation
Customization level	✓		✓		C1 C2 C3 C4	Heterogeneity of Customization levels (make, buyAndCustomize or buy) of a domain's applications
Business functions	✓			✓	C2	Heterogeneity of business functions supported by a domain's applications
Component categories		✓	✓		C1 C2 C3 C4	Heterogeneity of infrastructure components of a given component category (e.g. operating systems, databases, etc.)
Coupled domains	✓			✓	C1 C2 C3 C4	Heterogeneity of the functional domains a given domain is coupled with, whereas the coupling between domains is determined by their applications and information flows between them
Databases		✓		✓	C2 C3 C4	Heterogeneity of databases used by a domain's applications
Interface implementation (Application)	✓			✓	C1 C3 C4	Heterogeneity of the technical implementations of an application's information flows
Interface implementation (Domain)	✓			✓	C1 C3 C4	Heterogeneity of the technical implementations of the information flows of a domain's applications
Operating systems		✓		✓	C2 C3 C4	Heterogeneity of the operating systems used by a domain's applications
Operating system types		✓	✓		C3 C4	Heterogeneity of the operating system types (e.g., vendor and version) used by an application
Programming languages	✓		✓		C2 C4	Heterogeneity of programming languages the applications of a domain are based on

A = Application layer I = Infrastructure layer T = Components R = Relations

Table 5.5.: Selected heterogeneity-based indicators (Schneider et al., 2015c)

However, applications providing data or services to various other domains, e.g. the Data Warehouse (DWH), are expected to have higher functional heterogeneity regarding their coupled applications. Domains providing corporate management services, e.g. human resources, accounting or risk management, should have an average amount of heterogeneity regarding their coupled domains, because they need to exchange data more than value creating domains but less than DWHs. Due to the fact that Cases 1, 2 and 3 originate from the banking sector their domain models are very similar. Therefore, on the highest level a mapping of the differently named domains is possible. Based on such corresponding general domain model the *Coupled domain* heterogeneity values are calculated and summarized in Table 5.6. Therein, the numbers equivalent entropy measure (EM_A) denotes the calculated heterogeneity of the domain's coupled domains which allows for cross-domain comparisons. Furthermore, for each domain and case the number of characteristics (NC), i.e. the number of different domains that domain is coupled with via information flows between its business applications, is provided. The number of actual information flows is indicated by the number of entities (NE) column.

	C1			**C2**			**C3**			
Domain	EM_A	NC	NE	EM_A	NC	NE	EM_A	NC	NE	**Expected coupling heterogeneity**
Loans	7.9	13	153	5.15	9	213	18.69	28	588	low
Sales	7.96	10	71	7.86	11	129	16.16	27	285	low
Legal audit	5.59	11	112	3.2	5	34	13.52	28	203	medium
HR	3.39	4	10	4.95	6	17	10.44	24	174	medium
Controlling	7.8	12	124	8.73	12	67	12.99	30	1069	medium
Risk	4.52	6	40	5.81	8	62	5.77	27	1124	medium
DWH	6.84	20	90	9.23	14	249	15.75	29	946	high
Number of domains		13			16			33		

EM_A = Numbers Equivalent Entropy Measure
NC = Number of Characteristics NE = Number of Entities

Table 5.6.: Coupled domain heterogeneity indicator results (Schneider et al., 2015c)

By analyzing the EM_A values of Table 5.6 we can see that each application landscape is amendable regarding its compliance towards the target state description provided in the last column. In general, the DWH domain has a higher coupled domain heterogeneity compared to the others. In Case 2, it actually has the highest value as expected by the target state description. While the *Loans* domain has a relatively low EM_A value—as expected—the value generating domain sales has a relatively high value in Case 2. This can be regarded as a domain of interest for the enterprise architect because here the reduction of heterogeneity might be appropriate. A positive example can be found in the *Risk* domain of Case 3. As expected, this domain has a low to medium heterogeneity of coupled domains. By comparing the number of coupled domains (NC) with the others this fact might be kept hidden from the enterprise architect. Counting information flows (NE) would also yield to other results which might be misleading.

When confronted with their individual indicator results of all indicators introduced and described in Table 5.5, the respective enterprise architects agreed that these results match with their expectations. It should be highlighted that the enterprise architect of Case 1 was surprised about one application which was ranked as one of the most heterogeneous applications with respect to information flows (*Interface implementation*). It turned out that this specific application which has been neglected by the enterprise architect up to this point in time was a DWH. The reason why it has been neglected during heterogeneity considerations was that it is classified as an IT application not as a business application. Therefore, it can be concluded that indicators in general and the heterogeneity-based indicator proposed by Schuetz et al. (2013) in particular assist enterprise architects due to their objective nature. Although this indicator has a strong theoretical foundation and is clearly defined its management suitability is—according to the participating enterprise architects—limited because the calculation description is quite difficult to understand (it contains logarithms and a sigma sign). In addition, indicator results cannot be aggregated, e.g., from sub-domains to domains which aggravates understandability. As illustrated in Table 5.3, not all companies were able to provide the required data for this indicator. Therefore, it can be considered to cause an additional data collection effort at least for some companies. Accordingly, indicator assessment results are visualized in Figure 5.6.

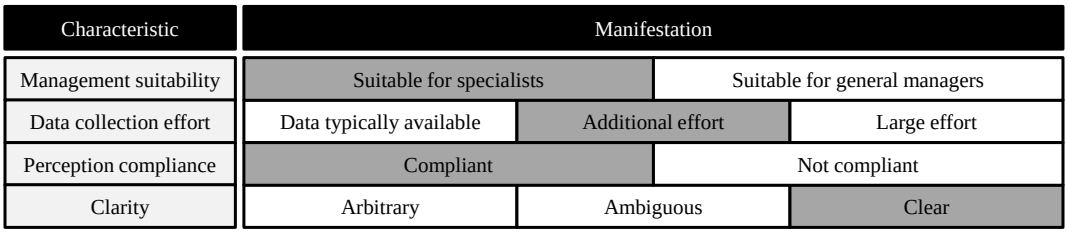


Figure 5.6.: Heterogeneity indicator assessment results

5.3.3.3. Topology indicator evaluation

The indicators proposed by Lagerström et al. (2014) are based on the topology of the application landscape which is determined by applications and their information flows. The approach consists of two parts: a clustering algorithm and the indicator *Propagation cost*. The clustering algorithm creates four distinct clusters of applications: *core* applications, *control* applications, *shared* applications and *periphery* applications. Thereby, the *core* and *shared* applications are considered to be the most complex applications because changes to those applications might propagate through large parts of the application landscape resulting in unforeseen efforts. The *Propagation cost* indicator yields the amount of possibly affected applications if a randomly chosen application is changed. In order to assess the indicators' utility the clustering algorithm as well as the propagation cost indicator have been applied to all four real-world EA descriptions available. The results are depicted in Table 5.7. Because this approach has never been applied to an entire application landscape before, the first question of interest is the indicators' plausibility. The calculation results indicate that all four application landscapes under investigation share some similarity in terms of cluster sizes. For example, the application landscape *core* varies between 37% and 55%. It is interesting to see that the core forms the largest cluster in nearly

Cluster / Indicator	C1	C2	C3	C4
Core	38 %	55 %	45 %	37 %
Control	14 %	7 %	7 %	17 %
Shared	8 %	11 %	13 %	3 %
Periphery	40 %	27 %	35 %	21 %
Propagation cost	24 %	41 %	30 %	21 %

Table 5.7.: Topology-based indicator results (Schneider et al., 2015c)

all cases followed by the periphery. Therefore, it can be concluded that the clustering algorithm yields sensible results across organizations. A deeper analysis of the results reveals some significant differences among the four cases. For example, the largest *core* consists of 55% (Case 2) while the smallest *core* consists only of 37% (Case 4). Therefore, on a relative scale the largest core is about 50% larger than the smallest which is an indicator of higher complexity. An even greater difference can be found in the comparison of the different propagation costs. The calculated propagation cost in Case 2 is about two times the propagation cost observed in Case 4. Therefore, changes to applications in Case 2 are much more likely to create unforeseen additional changes compared to Case 4. Again, the interviews with respective enterprise architects revealed that in all cases applications which are perceived to be highly complex are clustered either as *core* or as *shared* application while applications having low perceived complexity are clustered as *control* or *periphery* application. The data necessary to calculate the *Propagation cost* indicator and to apply the clustering algorithm consists only of applications and their information flows. This is fundamental EA data which was available in all four cases. The definition of the calculation rule is unambiguous and leaves no room for speculations. However, the enterprise architects have been doubtful regarding the indicator's suitability to actually manage or steer and application landscape towards the intended goal. Therefore, the indicator is only suitable for specialists not for general managers. Nevertheless, the indicator might be appropriate to prioritize change projects with respect to their potential change propagation. The results of this indicator assessment are visualized in Figure 5.7.

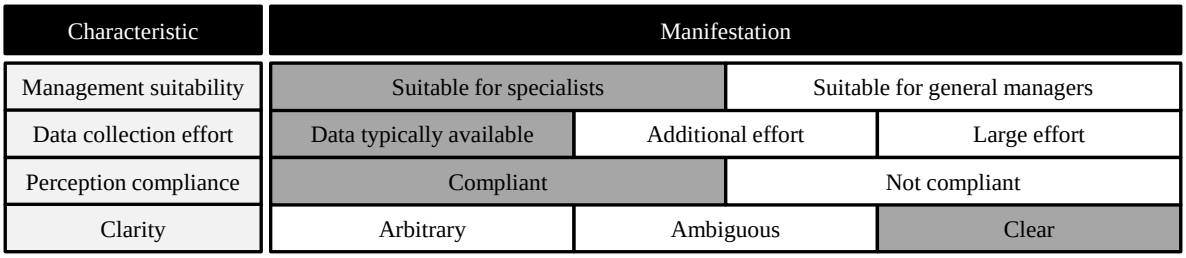


Figure 5.7.: Topology-based indicator assessment results

5.3.3.4. Indicator correlation analysis

When indicators are presented to higher-level executives for decision support usually their number should be minimal in order to avoid the inclusion of irrelevant data. To reduce the number of indicators required to achieve a holistic view on the complexity and diversity of application landscapes correlations between the results of two indicators can indicate a dependence of their outcomes. Therefore, correlation coefficients have been calculated for all pairs of indicators in all four use cases introduced before. Because for most pairs no correlation could be identified focus will lie on those having high correlation coefficients. By using the indicator results as input the statistical software R (see R Core Team, 2014) has been used to calculate respective Pearson correlation coefficients. This coefficient takes values between -1 and 1 and describes the degree of a linear dependence between two variables. Here, a high dependence is assumed, if the correlation coefficient is greater than 0.4 for two complexity indicators. Table 5.8 depicts respective correlation coefficients as well as their significance according to a two-sided t-test based on the following schema: * indicates a p-value <0.05, ** indicate a p-value <0.01 and *** indicate a p-value <0.001.

	Total number of information flows	**Total functional scope**
Number of applications	C1: -0.08 C2: 0.71 ** C3: 0.75 *** C4: 0.68 ***	C2: 0.82 *** C3: 0.63 *** C4: 0.97 ***
Total functional scope	C2: 0.61 * C3: 0.63 *** C4: 0.73 ***	-

Table 5.8.: Industry indicator correlations and their significance (Schneider et al., 2015c)

As can be seen in Table 5.8, the indicators *Number of applications*, *Number of Information flows* and *Functional scope* strongly correlate with each other on a very significant level for domains in Cases 2, 3 and 4. Therefore, it can be concluded that the more applications are part of a particular domain, the more information flows and function points this domain has. While this result is not surprising, it contributes to the plausibility of these indicators. Nevertheless, it needs to be highlighted that such correlation could not be found in Case 1 for which only one correlation could be calculated because of missing data for the other indicators. However, because correlation does not imply causation (Wright, 1921), a deeper inspection of the observed and strongly significant correlation is required.

In addition, Table 5.9 summarizes the relevant correlation coefficients for the identified cluster of heterogeneity-focused indicators. By analyzing the correlation coefficients in Table 5.9, one can see strong evidence for a correlation between the *Operating system heterogeneity*, *Database heterogeneity* and *Customization level heterogeneity*. Therefore, we can conclude that if the heterogeneity of customizations within a domain is very high, it is likely that the operating systems and databases used in this particular domain are also heterogeneous. Again, this observation needs to be used for future research about reasons causing this correlation.

	Operating system heterogeneity	**Database heterogeneity**
Customization level heterogeneity	C2: 0.67 ** C3: 0.29 *** C4: 0.68	C2: 0.69 ** C3: 0.43 *** C4: 0.62 ***
Database heterogeneity	C2: 0.84 *** C3: 0.61 *** C4: 0.31 *	-

Table 5.9.: Observed heterogeneity-focused indicator correlations (Schneider et al., 2015c)

To determine the correlation between heterogeneity-based as well as observed industry indicators and topology-based indicators logistic regression models have been used because the topology-based indicator is not interval scaled. Although there exist some significant correlations, e.g. a p-value <0.001 for the *Functional scope* and the application clusters in Case 3, no empirical evidence for a general correlation between heterogeneity-focused or observed industry indicators and topology-based indicators could be found. Therefore, it seems that they measure independent aspects of application landscape complexity.

5.3.3.5. Indicator assessment summary

In the previous chapters the indicator assessment framework developed in Chapter 5.3.1 has been used to assess the utility of all complexity and diversity indicators applicable for application landscapes which have been either observed in practice (see Chapter 5.1) or identified via a structured literature review (see Chapter 5.2). The assessment is based on the actual calculation of these indicators in four different real-world use cases, subsequent interviews with corresponding enterprise architects and a group discussion during a half-day workshop in 2014. Based on the individual assessment results visualized in Figures 5.5, 5.6 and 5.7, the following conclusions can be drawn:

- Indicators observed in industry are better suited for general managers than heterogeneity or topology-based indicators. Nevertheless, heterogeneity and topology-based indicators provide useful insights for specialists.
- Non of the analyzed indicators requires huge data collection efforts. For most indicators the data is typically available from EA repositories. Only the heterogeneity-based indicator might cause some additional data collection effort because it requires detailed information about specific attributes which might not yet be documented.
- Surprisingly, all indicators were able to identify highly complex applications according to respective enterprise architects' perceptions.
- Only the heterogeneity and topology-based indicators are based on a sound and clear definition. Indicators observed in industry (especially the *Customization level* indicator) are defined in an ambiguous fashion which could not be resolved because there was no consensus about the impact of customized software and customized products.

- Strong significant correlations have been found between indicators *Number of applications*, *Number of Information flows* and *Functional scope*. Therefore, one of those indicators might be enough as decision support.
- No correlation could be found between the different types of indicators. Therefore, it seems that they all measure different aspects of complexity and diversity.

In accordance with the literature review, the enterprise architects participating in this indicator evaluation were missing a context sensitive measure of diversity which accounts for the distance between two types of interest and therefore provides concrete decision support regarding standardization questions.

5.4. Metric selection and development

The analysis of complexity and diversity indicators currently used in practice (see Chapter 5.1) as well as documented in literature (see Chapter 5.2) resulted in the identification of various indicators with different benefits and drawbacks (see Chapter 5.3). Compared to the conceptual understanding developed in Chapter 4 these indicators only cover the *variety* and *balance* aspects of diversity. Neither available literature nor experienced practitioners provide a metric for the *disparity* aspect. Therefore, such metric is developed and presented in this chapter. In addition, an indicator describing balance not only between different products but also among different versions of a single product is presented; thus overcoming current restrictions to nominal scales.

5.4.1. The Goal-Question-Metric approach

A common approach to develop (software) metrics is the Goal-Question-Metric approach (Basili and Weiss, 1984; van Solingen and Berghout, 1999). It uses goal-directed data collection to evaluate methodologies with respect to the claims made for them. The initial goal of the Goal-Question-Metric approach was to illustrate how to obtain valid data that may be used both to learn more about the software development process and to evaluate software development methodologies in production environments. Nevertheless, the approach proved to be suitable also in other domains, e.g., business process model analysis (Ghani et al., 2008). In total, the Goal-Question-Metric approach consists of six consecutive steps (see Figure 5.8).

Within the first step, the goals of the data collection are established. Such goals could either focus on the evaluation of a methodology relative to the claims made for it or be relevant for all methodologies. This step ensures that the obtained data includes the patterns one is looking for. In the second step, a list of questions of interest is developed based on the defined goals. These questions are used to identify data parameters and categorizations. If questions of interest are not or cannot be formulated, goals are not well defined. Once the questions have been established, categorization schemes may be constructed. Generally, each question induces a categorization scheme. In order to collect the required data for calculating metrics and answering the established questions of interest a data collection form is developed in step four. Such data collection form has to solve the conflict between the desire to gather a complete, detailed set of data and the need to minimize the time and effort involved in supplying this data. In the

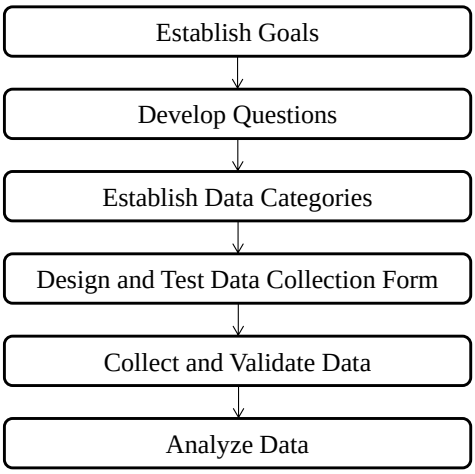


Figure 5.8.: The Goal-Question-Metric approach according to Basili and Weiss (1984)

best case, data suppliers are involved in the form design process. With such data collection form at hand in step five the actual data collection can take place. Validation of collected data consists of checking the forms for correctness, consistency and completeness. Within the final step, the collected data is analyzed by calculating metrics to answer the established questions of interest. If necessary, the Goal-Question-Metric approach allows iterating among the steps several times (Basili and Weiss, 1984).

The first step of the Goal-Question-Metric approach (see Basili and Weiss, 1984) is to define the goal of data collection and analysis. To specify goals in a structured way the GQM templates proposed by Caldiera et al. (1994) can be used (van Solingen and Berghout, 1999). The GQM template requires to formulate a sentence of the following structure:

Analyze	the object under measurement
for the purpose of	understanding, controlling, or improving the object
with respect to	the quality focus of the object that the measurement focuses on
from the viewpoint of	the people that measure the object
in the context of	the environment in which measurement takes place

Table 5.10.: GQM template according to Caldiera et al. (1994)

Based on this template, the goal for which metrics should be developed here can be defined as follows:

Goal:
Analyze the application landscape description for the purpose of controlling its evolution with respect to its diversity from the viewpoint of an enterprise architect in the context of a specific enterprise.

Based on this established goal, the next step requires the derivation of questions of interest.

These questions will serve as a basis for data categorization later on. Therefore, the object of analysis, i.e. an application landscape description, needs to be decomposed into relevant parts to be analyzed. Such categorization can be achieved by referring to the constituting parts of an application landscape as already described in Chapter 4.1 as well as the typical interest of today's enterprise architects as identified in Chapter 5.1. The former include business applications on the one hand and infrastructure components on the other hand. The latter distinguishes different types of clusters of infrastructure components, e.g., database systems, operating systems or web servers. In addition, the diversity framework presented in Chapter 4 provides the dimensions of interest regarding the previously established goal. These include variety, balance and disparity. Based on the empirical observations described in Chapters 3 and 5.1, the scope of the questions of interest can be limited to specific domains instead of considering the whole application landscape at once. Accordingly, the following questions of interest can be derived for both, the application landscape scope or a domain-centric scope:

- How many different types of business applications belong to the application landscape?
- How are business application instances distributed among the different types associated to the application landscape?
- How different are the types of business applications belonging to the application landscape from each other?

In addition to business applications, technical components, e.g., database management systems and operating systems, can also be assessed. Most of the data categories which can be derived from these questions have already been stated above. Still missing are the criteria which are used to define *difference* among types. These criteria are of course type-specific, i.e. the difference among database systems is determined differently from the difference among operating systems. Furthermore, these criteria might also be company-specific. Therefore, it is refrained from providing a common and complete set of criteria here. Instead, concrete examples will be presented in Chapters 5.4.4, 7.3 and 7.5.

5.4.2. Reusing metrics for variety and balance quantification

To assess *variety* in a set of (EA) elements the number of used concepts needs to be calculated (Stirling, 2007; Page, 2011). Respective indicators have already been described in literature and are also used in practice. For example, Schuetz et al. (2013) count the number of database products used within an application landscape as part of their heterogeneity considerations, Mocker (2009) counts the number of operating systems per application and four out of six companies count the number of infrastructure components used to realize a business application (Schneider et al., 2015c). In each case, several concepts are used to describe a set of individuals, e.g, the name of a database product is used to describe all installations of this product. Furthermore, a concept exists for the union of these concepts, e.g., database. Therefore, it can be concluded, that counting the number of concepts used to describe a set of individuals which all belong to a more general concept is already a suitable indicator to quantify *variety*.

To quantify *balance*, not only the number of concepts needs to be regarded but also the distribution of individuals (Stirling, 2007; Page, 2011). Therefore, entropy measures qualify for the

quantification of balance. Literature provides a large set of entropy or concentration measures including the Herfindahl Index (Herfindahl, 1950), Horvath Index (Horvath, 1970) and Shannon Entropy (Shannon, 1948). To select an appropriate measure for balance quantification in the context of EAM, Schuetz et al. (2013) provide an analysis of these measures with respect to shifts in classification and the effect of small sets or elements and conclude that Shannon Entropy is best suited. However, this approach allows only to assess balance if there is no relationship between the concepts under investigation which needs to be considered. For example, a specific database product might be used in different versions. Therefore, time provides a natural order of these versions. Because entropy and concentration measures work on nominal scales they cannot capture such ordinal scale and therefore their decision support is limited in such cases. In the following chapter a suitable measure is presented closing this research gap.

5.4.3. A novel metric for balance quantification

As outlined before, *balance* is a function of the pattern of apportionment of individuals across concepts. To quantify balance one needs to be aware of the scale used to describe the distribution of individuals. According to Stevens (1946), the scale could be either nominal (no relationship between concepts), ordinal (a total order of concepts is known) or interval (degree of difference between concepts is known). The aforementioned measure proposed by Schuetz et al. (2013) which is based on Shannon Entropy (Shannon, 1948) disregards any relationship between concepts. Therefore, it is only suitable for nominal scales. Accordingly, it falls short in describing balance in case a natural order of concepts exists. Such case can be found, for example, when elements of an application landscape exhibiting adolescence should be identified. According to Schneider et al. (2016), adolescence can be identified by analyzing the distribution of individuals among the different versions of a product in use. The more old versions are in use, the greater the adolescence of the respective product. In this case, the release cycle of the product defines a natural order of its versions. Accordingly, balance measures based on nominal scales fall short in characterizing the pattern of distribution. A concentration on the oldest version means something different than a concentration on the newest version, but to express this fact concepts need to be treated ordinal. In the following, a respective measure proposed by Schneider et al. (2016) able to express the pattern of distribution also for ordinal scaled data based on skewness is presented.

The term *adolescence* can be understood in different ways. In the following, adolescence will be defined for internal or external business applications and technical components which are subject to release management and therefore a controlled lifecycle. In case of external components, this lifecycle is fully controlled by the respective vendor. Adolescence becomes visible, for example, if a specific version runs out of maintenance support. Accordingly, it can be defined as follows.

Definition: Adolescence of IT components
An IT component ages due to the availability of newer major or minor versions of the same product as well as the availability of a succession product of the same vendor. The degree of adolescence of a an IT component is thus the skewness of the distribution of its instances among its versions.

(Schneider et al., 2016)

In the following, papers addressing subjects similar to the adolescence concept are presented to provide an overview of the scientific knowledge.

Since more than two decades, researchers examine legacy information systems (Kardasis and Loucopoulos, 1998). Although age is not a sufficient condition, most legacy systems are characterized by a high age. In this context, age is not limited to the software itself but can also apply to the respective hardware. Common issues in the context of legacy systems include, but are not limited to, slow and expensive hardware, high maintenance costs and difficulties during integration with other IS (Bisbal et al., 1999).

In the context of IT controlling, the lifecycle of business applications is commonly concerned. This lifecycle can be divided into the phases planning, initial development, production, extension and retirement (Zarnekow et al., 2004). Thereby, the total age of such application is determined as the time-span between the beginning of planning and the end of its retirement, or the current date for applications being still in production. The extent of enhancements and changes carried out during its lifetime is therefore usually neglected.

In the course of searching for application landscape complexity drivers, Mocker (2009) identified the age of business applications as a major influence factor. Thereby, age has been defined as the number of years since the first go-live day of a certain business application. Again, the extent of maintenance carried out is not considered. In addition, no relationship between age and other aspects like functional redundancy or deviation from standards could be observed.

The amount of time a certain IT component is used, i.e. its age, has impact on the usage of new technologies due to path dependence (Fürstenau and Kliewer, 2015). The so-called lock-in effect is also described by Arthur (1989) and shows how difficult the migration of old technologies to new technologies can be.

An intuitive understanding of age in the context of application landscapes can be found in Hanschke (2014). Age, among other aspects, is here used to describe the current *health condition* of a business application. Thereby, functionally similar applications can be compared easily. The adolescence of an application landscape is thereby mentioned as a potential problem although a detailed elaboration of age is missing.

Conceptualizing adolescence of IT components

Determining the age of an IT component based on the day of introducing the component in the application landscape falls short in many respects. On the one hand, an IT component, e.g. a business application, is mostly composed of other components (The Open Group, 2011) which

are subject to their own lifecycle raising the question of how to aggregate the age of different components. On the other hand, IT components are subject to maintenance activities (Zarnekow et al., 2004) which raises the question of the meaning of age in such context. In addition, especially for COTS products, the day of introducing the component in the application landscape cannot be considered to be equal to the day of finishing production of this component. Therefore, it can be much older. After finishing the initial development, often continuous development takes place. Thereby, intermediate versions are released after certain time-intervals (Ramakrishnan, 2004). An analysis of such product reveals that it can change significantly during its lifecycle and a new version could sometimes be compared to a completely new product.

An illustrating example can be found in the operating system developed by Microsoft. Using the initial release as kind of birthday would imply that an installation of the recently released *Windows 10* would be 30 years old due to the fact that *Windows 1.0* has been released on November, 20^{th} 1985. When instead the day of introducing a component into the application landscape is used to denote a component's birthday, an installation of *Windows XP* in 2006 would be regarded to be one year old in 2007 although *Windows XP* has been released already in 2001. In fact, the component should be considered aged when it was first installed in 2006.

Although a business application can be distinguished clearly from IT components such as operating systems or database management systems, this is not trivial for other IT components such as libraries or programming languages. Again, this rises the question how the age of such components influences the age of the business application. In addition, comparing the age of different IT components is difficult due to fact that their release cycles might differ significantly. The faster the development of a certain component proceeds, the faster it ages and becomes adolescent. Focusing solely on time, this aspect is not addressed adequately. Therefore, a conceptualization of age and adolescence is required allowing quantification and comparison as well as application to all IT components.

Release management forms a common approach to manage dependencies between releases of IT components and to ensure stability of all its elements (Ramakrishnan, 2004). In the sense of an application landscape, both business applications and infrastructure components are subject to a specific release cycle. This release cycle could be managed either exogenous (by the product vendor) or endogenous (by the company itself). In contrast to the time since the introduction of a component, the release cycle clearly defines a predecessor-successor-relationship which forms a stable foundation for the consideration of IT component's age. However, this implies the usage of an ordinal scale, because the intervals between single releases must not be equal, independently of considering time, functionality or other criteria for comparison. The frequency of certain releases of the same IT component within an application landscape can be mapped to a distribution function by using the predecessor-successor-relationship. The shape, i.e. the location, of such distribution function in turn provides hints regarding the adolescence of the respective IT component. Accordingly, location parameters like median, mean, standard deviation and skewness as measures for the aggregation of the distribution function's location qualify for adolescence considerations.

Indicator design

A measure of adolescence has to fulfill certain additional requirements. For example, the naive approach to determine the age for each release/version, calculate the average value (maybe weighted by the frequency of instances) and compare it to a target value should, according to the aforementioned reasons, not be used. Especially the determination of a target value cannot be achieved in general for arbitrary IT components and a specific determination requires additional effort and profound knowledge about each IT component. Instead, by using relative frequencies of release instances, a distribution function can be derived which can be described on an aggregated level by location parameters. To define a suitable indicator quantifying adolescence of IT components, such location parameter needs to fulfill the following characteristics (Schneider et al., 2016):

1. The more old versions of an IT component are used, the higher the adolescence.
2. If a certain instance is migrated to a newer version, the adolescence decreases.
3. Older versions have stronger influence on adolescence than less older versions.
4. The ratio of instances per version determines an IT components adolescence, not the absolute number of instances.
5. The distribution function requires only the predecessor-successor-relationship between versions, no distances.
6. The availability of a new release, even if not part of the application landscape yet, yields an increase of the IT component's adolescence.

Requirements 1 and 2 are derived directly from the previously described definition of adolescence. Requirements 3 and 4 are commonly desired properties of indicators like measures of concentration which describe distributions (Hall and Tideman, 1967; Shapiro, 2010). Requirement 5 reflects both, the limitations regarding commonly available data and the difficulty of defining the age of a certain release absolutely. For example, it is conceivable that a DBMS is released in version 5.8 and then new features are released in version 6.0. Because the migration to version 6.0 might require extensive effort, the respective vendor might release version 5.9 after version 6.0 to fix some software errors. Accordingly, the chronological order of these versions does not mirror the typical migration path for companies. Solely the predecessor-successor-relationship provides information about the order of certain releases. Therefore, the indicator to be developed has to be computable based on ordinal data. Assuming interval-scaled data is not possible because neither are the time-intervals between releases equal nor is the effort required for migration equal. Requirement 6 reflects the impact of the technological progress and ensures that the adolescence of an IT component increases if new versions are released.

Among the possible location parameters for describing the distribution function of releases, especially the skewness parameter qualifies for capturing adolescence of IT components. Thereby, different skewness definitions can be used. Table 5.11 provides an overview of commonly used skewness parameters including their respective calculation rule based on the works of Handl (1985) and Klein (1999).

Skewness	Origin	Calculation	Foundation
SO_I	Klein (1999)	$-(k-1)+2\sum_{i=1}^{k-1} F_X(x_i)$	Values of the distribution function
$b_2(a)$	Groeneveld and Meeden (1984)	$\frac{F^{-1}(0.25)+F^{-1}(0.75)-2F^{-1}(0.5)}{F^{-1}(0.75)-F^{-1}(0.25)}$	Median and quantil distances
S	Oja (1981)	$\frac{E(X)-F^{-1}(0.5)}{\sqrt{Var(X)}}$	Median and standard deviation
MC_n	Brys et al. (2004)	$med_{x_i \leq m_n \leq x_j} \frac{(x_j-m_n)-(m_n-x_i)}{x_j-x_i}$	Median and values of the distribution function

Table 5.11.: Common skewness measures found in literature

Each of the aforementioned skewness measures satisfies requirements 1 to 4. Requirement 6 is in the narrower sense not fulfilled by any skewness measure but can be satisfied by adapting the data input accordingly. Thereby, when calculating means or standard deviations, newer versions have to be added as characteristic attribute with frequency 0. Because only the skewness measure proposed by Klein (1999) is able to calculate skewness for ordinal scaled data (requirement 5) the other measures disqualify due to their assumption of equal distances between the different versions of an IT component. Accordingly, this measure satisfies all requirements and is therefore selected to calculate adolescence of IT components. The general computation rule is defined as follows. Thereby, k denotes the number of characteristics, i.e. the number of versions of the respective IT component. $F_X(x_i)$ denotes the expected value for an instance to be of characteristic x_i or a smaller characteristic which can also be denoted as $P(X \leq x_i)$. In other words, this represents the accumulated relative frequency of characteristic x_i.

$$-(k-1)+2\sum_{i=1}^{k-1} F_X(x_i) \tag{5.18}$$

Given the exemplary distribution depicted in Figure 5.9, the adolescence of the Windows component can, according to Equation 5.18, be calculated as follows.

$$-(6-1)+2\left(\frac{30}{300}+\frac{110}{300}+\frac{121}{300}+\frac{202}{300}+\frac{292}{300}\right)=-0,027 \tag{5.19}$$

The negative sign in combination with a small absolute value shows that the distribution is slightly left skewed implying that a little bit more new versions are in use. If the result is divided by $k-1$, the measure is transformed to a superior representation whose results are normalized between -1 and 1. The resulting measure satisfies also the requirement that arbitrary IT components can be compared with regard to their adolescence independently of the amount of different versions in use.

When calculating adolescence, all releases of a certain IT component in use have to be considered.

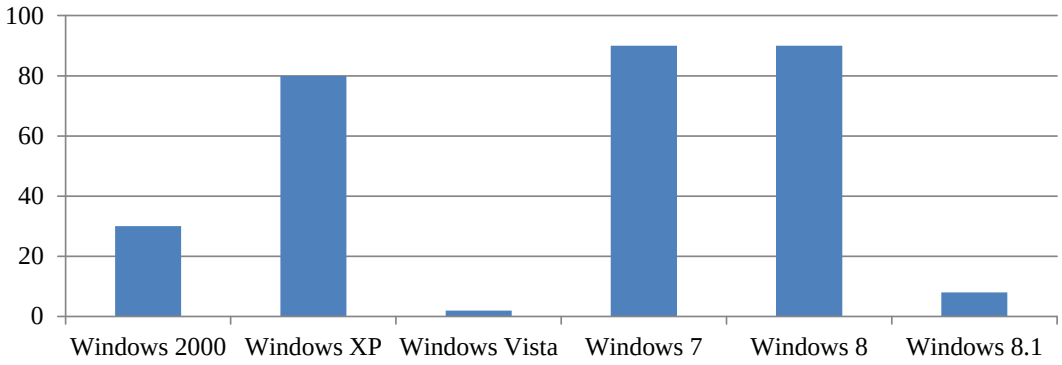


Figure 5.9.: Fictitious distribution of operating system instances with respect to their versions (Schneider et al., 2016)

Older releases which were never used or are not used anymore, do not need to be considered. The smallest characteristic is therefore the oldest actually used version of an IT component. If an organization uses more than one version, it is possible, that a version available on the market has been skipped. Such version has to be considered only, if it is part of the migration path of older versions. If there is no plan to use such version at all, it can also be skipped as characteristic. The largest characteristic is determined by the newest version used by the organization or the most recent version available on the market if using this version is considered possible and sensible. Figures 5.10 and 5.11 illustrate the behavior of the indicator in case of extreme left or extreme right skew by using absolute frequencies. In the left case, the indicator will result in 1 indicating the highest possible adolescence. In the right case, the indicator will result in nearly -1 indicating the absence of adolescence.

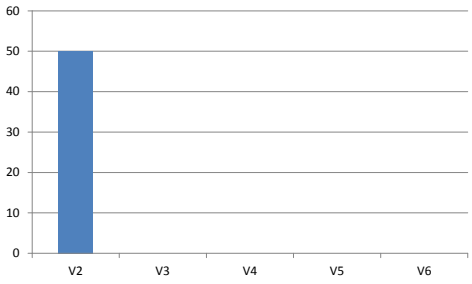


Figure 5.10.: Extreme right skewed distribution

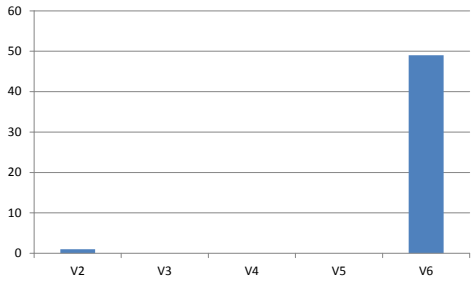


Figure 5.11.: Extreme left skewed distribution

Indicator visualization

For communication purposes and to increase the comprehensibility of application landscape properties, visualizations are frequently used (Hanschke, 2014; Lankes et al., 2005). Due to the fact that interpreting the results of the previously introduced indicator to quantify adolescence of IT components requires knowledge about its value range as well as the meaning of the minus-sign, an intuitive visualization is developed and presented in the following. Commonly used

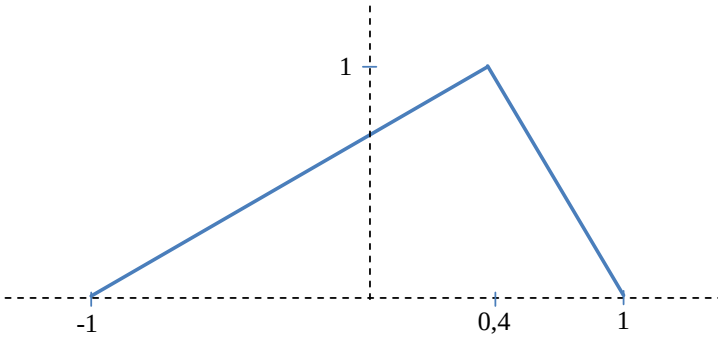


Figure 5.12.: Exemplary visualization of an adolescence value of -0.4 (Schneider et al., 2016)

box-plots seem to be inappropriate here, because they also require detailed knowledge about quartiles and their construction. Histograms can be interpreted without additional knowledge but do not provide an aggregated perspective on the distribution, thus limiting the ability to comprehend adolescence instantly. Therefore, a visualization based on the skewness of the distribution function is used to visualize adolescence directly. The visualization is based on characteristics of histograms, but reduces them to a simple triangle capturing only relevant information. The construction of the triangle begins by placing a horizontal line which will be removed afterwards. The edge opposing this line is placed based on a fixed distance as well as a mapping of the interval of potential indicator results [-1;1] to the length of the longest side of the triangle which resides on the line already drawn. Thereby, the sign has to be negated. Figure 5.12 shows an exemplary visualization for an adolescence value of -0.4 including also the axes used for construction.

In contrast to a histogram visualization containing the actual relative or absolute frequency for each version in use, the presented visualization indicates only the distribution of instances among different versions according to their skewness. Thereby, additional details are consciously omitted to increase the comparability of adolescence values for different IT components. By going without the coordinate system, a schematic visualization of adolescence is created. A detailed example with real-world data for both, the indicator and the visualization, is provided in Chapter 7.4.

5.4.4. A novel metric for disparity quantification

In the context of EAM, measuring *disparity* among concepts is currently unaddressed in literature (see Chapter 5.2). In general, distance measures are appropriate to quantify disparity among concepts (Page, 2011). In particular, as proposed by Stirling (2007), the metric presented by Weitzman (1992) is suitable to quantify disparity. Therefore, the remainder of this chapter develops and presents a novel metric based on Weitzman (1992) to assess disparity in the context of EAM.

The approach presented by Weitzman (1992) to quantify disparity uses a given distance function $d(x, y)$ where $x, y \in S$ and S is the set of all species (concepts). Thereby, the distance $d(x, y)$

stands for the number of (possibly weighted) character-state differences between elements x and y. This distance function has to satisfy the following conditions:

$$\begin{aligned} \text{non-negativity: } & \forall x, y \in X : d(x, y) \geq 0 \\ \text{identity: } & \forall x, y \in X : d(x, y) = 0 \Leftrightarrow x = y \\ \text{symmetry: } & \forall x, y \in X : d(x, y) = d(y, x) \end{aligned} \tag{5.20}$$

Note that these conditions equal those already presented in Equation 5.1 without the condition of triangle-inequality which allows the usage of non-metric distances. Given such distance function disparity denoted as $V(S)$ is recursively defined as:

$$V(S) = \underset{x \in S}{max}\{V(S \setminus x) + d(x, S \setminus x)\} \tag{5.21}$$

Equation 5.21 can be interpreted as follows (Page, 2011).

1. Let S equal the empty set and set $V(S) = 0$.
2. Randomly chose a concept to add to S.
3. Chose a concept y of least distance to a member of S according to $d(x, y)$. Then increase $V(S)$ by the distance from y to S and add y to S.
4. If not all concepts belong to S go to Step 3.

The metric proposed by Weitzman (1992) shows some very appealing properties. For example, the metric is able to calculate the loss of disparity in case a specific concept and all its individuals become extinct. If exactly one concept out of n concepts has to perish, e.g. due to budget constraints, and there is a choice the most intuitive answer is that one of the two most closely related concepts should become extinct in order to preserve as much disparity as possible. However, Weitzman (1992) already stated that—depending on the context—for the same collection of objects using different distance functions might be appropriate. Therefore, the outcome and expressiveness of this metric is strongly determined by the definition of the particular distance function. If such distance function is based on differences in attributes (as suggested by Weitzman (1992) and Page (2011)) a suitable distance function has to be developed for each specific case. Although there might exist common attributes of general interest, each concept to be assessed with regard to disparity is likely to exhibit specific attributes which need to be taken into account. Accordingly, characteristics used to determine distance between two database systems differ from those used for operating systems. To exemplify the approach, a hypothetical example is presented in the following.

Demonstration example

To exemplify the approach of measuring disparity within application landscapes the metric's applicability is demonstrated for DBMSs. In previous analyses regarding EA diversity or complexity DBMSs have often been used to demonstrate new metrics (Mocker, 2009; Schuetz et al., 2013) and practitioners often collect respective data (Schneider et al., 2015c). Previous research

focused on product vendors as a distinguishing characteristic which falls short as a single input for a context-sensitive distance function. Therefore, a generic distance function for DBMSs is derived first based on some general differentiating characteristics. Although not being tied to a specific organizational problem this distance function can be used to demonstrate the benefits of quantifying disparity. The goal is not to propose a distance function suitable for all cases but to provide a reasonable example.

One fundamental characteristic of each DBMS is its paradigm. Most of the instances in operation today follow the relational paradigm developed by Codd (1970). Nevertheless, there is an increasing number of DBMSs based on other paradigms which are often summarized under the term NoSQL (Fowler and Sadalage, 2012). Another characteristic of DBMSs is their focus on different use cases. While On-line Transactional Processing (OLTP) DBMSs focus on ensuring ACID properties of database transactions (Gray, 1981), On-line Analytical Processing (OLAP) DBMSs focus on decision support use cases (Chaudhuri and Dayal, 1997). Especially in business environments the price of a product, i.e. DBMS, is an easily conceivable characteristic. On a general level it can therefore be distinguished between expensive products offered by major vendors, e.g. Oracle, Microsoft and IBM, and open source or free products like MySQL and PostgreSQL. Although there are generally more characteristics conceivable, only these three characteristics are used to define the distance of DBMSs here.

The respective distance function should be metric to allow the representation of concepts, i.e., DBMS products, as points in Euclidean space. If each of the aforementioned characteristics is considered as a capability, each concept can be represented by a vector of length equal to the number of capabilities n. According to the capabilities provided by each type of DBMS, this vector x is filled with ones and minus ones, assuming an equal weight of all capabilities:

$$x_i = \begin{cases} 1, & \text{if capability is provided} \\ -1, & \text{otherwise} \end{cases} \tag{5.22}$$

As denoted in Equation 5.23, the distance of two DBMSs x and y can then be defined as the Euclidean distance between them. Other distance functions like the Jaccard index (Jaccard, 1912) would be equally appropriate.

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{5.23}$$

Accordingly, the minimum distance between two concepts, ie.e DBMS, is zero which occurs if both concepts equal exactly regarding considered capabilities. If two concepts differ in every single capability the distance function maximizes at $\sqrt{4n}$ where n denotes the number of capabilities. This way, the introduced distance function can be considered as a measure of overlap between concepts. The more overlap, the smaller their distance. Furthermore, it satisfies the conditions described in Equation 5.20 and is therefore a valid distance function for the Weitzman metric. To see the contribution of disparity to diversity consider two different EA models named A and B with DBMS instances according to the distribution given in Table 5.12. If the number of different types, i.e. *variety*, is calculated both EA models yield the same result

DBMS	Model A	Model B
MongoDB	5	25
Oracle 11g	25	5
Microsoft SQL Server	70	70

Table 5.12.: EA models where varying diversity is not discriminated by conventional indicators

of three concepts. If the distribution of instances is taken into account to quantify *balance*, both EA models yield the same result again due to equal relative frequencies. However, taking *disparity* into account one can recognize that the products of Microsoft and Oracle share more characteristics than they do with MongoDB, e.g. the SQL interface and the relational model. Table 5.13 describes the vectors used for distance calculations.

DBMS	SQL	NoSQL	OLAP	OLTP	Expensive	Cheap
MongoDB	-1	1	1	-1	-1	1
Oracle 11g	1	-1	1	-1	1	-1
Microsoft SQL Server	1	-1	1	-1	1	-1

Table 5.13.: Hypothetical vectors to describe DBMS disparity

Because both models include exactly the same concepts, the disparity metric yields to the same result. As stated before, all three dimensions of diversity have to be considered in parallel. Representing the disparity as tree (dendrogram) and using the data of Table 5.13 an integrated view on the diversity of both models can be derived as visualized in Figure 5.13.

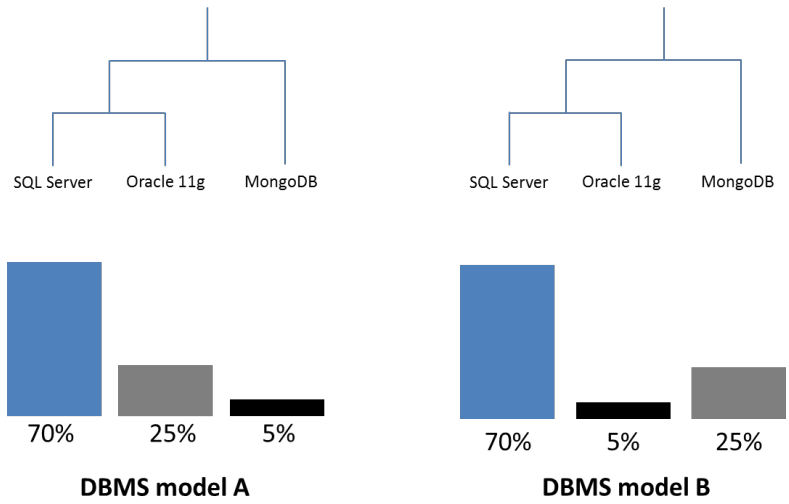


Figure 5.13.: Graphical representation of varying diversity discriminated only by disparity

This small hypothetical example demonstrates that the diversity of EA model B is higher than the diversity of EA model A because it consists of more individuals of the most disparate

concept, i.e. MongoDB. Therefore, the three dimensions of diversity need to be considered together because each dimension measured in isolation yields equal results in this example. Due to the fact that existing EA diversity indicators are not able to distinguish between the two (fundamentally different) models, the concept of disparity allows enterprise architects to view their technology portfolio from a new perspective thus providing better decision support for diversity management. However, "it is difficult indeed to contemplate any single general index of diversity that could aggregate properties of variety, balance and disparity in a uniquely robust fashion" (Stirling, 2007). In the following chapters the required data for calculating the previously introduced disparity metric is derived as required by Basili and Weiss (1984) followed by an empirical evaluation.

Data collection

In order to be able to calculate the disparity metric introduced before, required data needs to be collected. Although paper might be a usable medium (as initially suggested by Basili and Weiss (1984)), a software solution like an EA repository might also be suitable to collect the required data. Such tool allows, e.g., for distributed and repeated data collection which is automatically stored and therefore can be reused for other purposes easily. The conceptual model representing data required for disparity assessments is depicted in Figure 5.14.

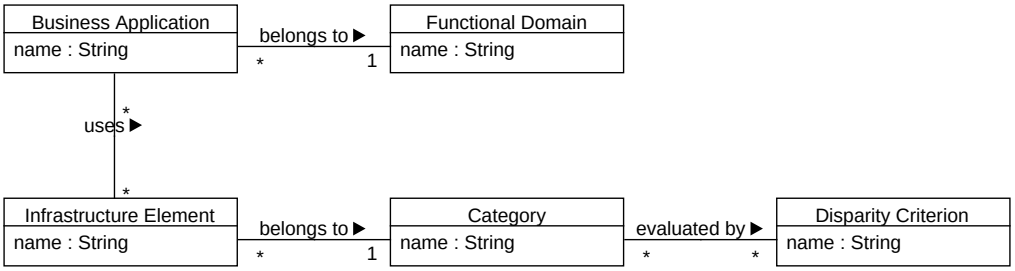


Figure 5.14.: Conceptual model for data collection

To assess disparity, concepts such as *Infrastructure Elements* representing different products need to be described by additional concepts, i.e., *Categories*. These categories could, for example, distinguish between Database Management Systems, Operating Systems and Web Servers. At this level, the concepts for which disparity should be assessed are defined. Subsequently, for each *Category* the desired characteristics for distance calculations need to be documented. Based on these criteria, the associated *Enfrastructure Elements* need to be extended with respective ratings. The relationship from *Infrastructure Elements* to *Business Applications* and *Functional Domains* is required to assess disparity also on the level of functional domains and not only for the whole application landscape.

Basili and Weiss (1984) suggest to test the developed data collection form before starting to actually collect data. Although each company will have to test a concrete instantiation of the previously introduced form on its own, its general applicability should be demonstrated here. Therefore, three different questions will be answered:

- Are companies able to collect the required data?

- Are tools available supporting data collection?
- Do enterprise architects have access to the required information to analyze it?

An analysis of current EA practice reveals that companies typically document their EA by modeling business applications and infrastructure elements which have been used to build them (Schneider et al., 2015c). In addition, infrastructure elements are usually classified to some sort of taxonomy, i.e. domains. Therefore, it can be concluded that companies are able to collect required data at least for DBMS and Operating System (OS), tools to store such data are available and enterprise architects have access to required data.

Empirical evaluation

To evaluate in how far the previously introduced disparity metric is able to provide relevant information to enterprise architects, the metric is applied to real-world EA descriptions and enterprise architects are interviewed accordingly. Therefore, this evaluation addresses both internal validity and external validity (Campbell et al., 1963). To demonstrate the additional facet of diversity added by the disparity metric, three real-world EA models which have already been described in Chapter 5.3.2 serve as a basis for practical evaluation.

Table 5.14 shortly recapitulates the origin of the data as well as its relevant characteristics. In each case a snapshot of the entire EA model from 2014 was available. For demonstration purposes the focus lies on the disparity of DBMS products used by these companies. Therefore, the number of types indicates the number of different DBMS products in use without considering different versions of these products (variation). The number of DBMS instances represents the number of business applications using one of these DBMS products. Therefore, it is not necessarily equal to the real number of operating instances but from an enterprise architect's point of view thoroughly the more interesting number. Furthermore, Table 5.14 provides the total number of business applications modeled by the respective enterprise architects.

Case	Industry branch	Company size	Headquarter	DBMS types	DBMS instances	Business applications
1	Banking	Large	Germany	8	1,144	1,898
2	Banking	Middle	Germany	10	185	247
3	Insurance	Middle	Germany	8	147	234

Table 5.14.: Description of disparity evaluation data

When asked about data quality, the respective enterprise architects answered that they are satisfied with this data and use it as basis for EA related decisions. To compare the three EA models described above two types of DBMS (and its corresponding instances) had to be removed due to different modeling approaches. Thus, the DBMS type called *Excel* with two instances and the DBMS type called *no data storage* from the model of Case 3 have been excluded because although present in the EAs of Cases 1 and 2 such instances have not been documented.

To calculate *variety* and *balance* indicators regarding DBMS types and instances in these three

cases the indicators proposed by Mocker (2009) and Schuetz et al. (2013) are used (see Chapter 5.4.2). *Disparity* is calculated using the metric proposed by Weitzman (1992) and the general distance function presented at the beginning of Chapter 5.4.4. Table 5.15 summarizes the results.

Case	Variety	Balance	Balance (norm.)	Disparity
1	8	1.653	5.227	8.485
2	10	1.568	4.795	8.485
3	8	1.122	3.074	11.314

Table 5.15.: Empirical DBMS diversity measurement results

By comparing Cases 1 and 3 one can see that although the number of DBMS products in use is equal, in Case 3 more disparity has been achieved. Likewise, if Cases 1 and 2 are compared, the disparity is equal but in Case 2 two more DBMS products are in use. Based on these observations, Case 3 seems to have lowest variety and balance but also exhibits the largest disparity and can therefore be regarded superior to the other cases.

In Case 1, the application landscape is partitioned by a three-layered model of functional domains. Due to the large extend of the application landscape, the respective enterprise architects manage diversity not on a global level but on a domain level. Therefore, variety, balance and disparity indicators are also calculated for each available domain. The results show many domains having equal variety and balance so disparity is analyzed especially for these domains. For example, the following domains showed a variety of 2.83 and a balance of 0.693 (which equals an equal distribution of 2 concepts) although having a different number of DBMS instances and different DBMS concepts: Compliance Services, Operational Risk, Product Conditions, Legal, Enterprise Application Integration, and Technical Services. Without considering disparity all these domains would be attributed to have equal diversity. Applying the basic distance function presented before different results can be obtained (see Table 5.16). To exemplify the approach of applying weighting factors for some capabilities and therefore adjust the distance function towards the strategic orientation of the company, Table 5.16 also shows the values of weighted disparity. Thereby, the possible Euclidean space has been extended. In this example, a difference in *use case* is considered to add more distance to two DBMSs than a difference in their *paradigm*. Consequently, the weighted disparity values for the *OLTP* and *OLAP* capability take values of either 2 or -2.

Regarding the disparity metric results, an enterprise architect can easily identify domains where standardization, i.e. reduction of diversity, can be performed although a maximum of disparity should be preserved. In the example given above the domain EAI is a good candidate for variety reduction. Reducing the number of DBMS products there comes with the minimal loss of disparity (while variety and balance losses are equal for all domains) because according to the defined capabilities the two DBMSs are not different. However, the metric results can only help in finding a spot to start deeper inspections. Technology components such as DBMSs might be tied to application components and diversity reduction might therefore not be feasible without changes on other architectural levels. However, with regards to internal validity it can be concluded that the analyzed disparity metric provides additional information and behaves as

Domain (instances)	Variety	Balance	Disparity	Disparity (weighted)	Concepts
Compliance Services (4)	2	0.693	2.83	2.83	MS SQL, Oracle
Operational Risk (2)	2	0.693	2.83	4	Sybase, Oracle
Product Conditions (4)	2	0.693	2.83	2.83	DB2, IMS
Legal (2)	2	0.693	4	4.90	MS SQL, Sybase
EAI (2)	2	0.693	0	0	DB2, Oracle
Technical Services (2)	2	0.693	2.83	2.83	Oracle, IMS

Table 5.16.: Extract of the diversity measurement results for Case 1

expected. To ensure also external validity interviews with enterprise architects are performed an described in the following.

To evaluate to what extend practicing enterprise architects value the proposed disparity metric for supporting diversity management twelve interviews in nine sessions are conducted. Table 5.17 describes the background of the interviewees. All interviews have been conducted either face-to-face or via telephone and lasted between 45 and 78 minutes.

Id	Role	Industry	Experience
1	Global Chief Enterprise architect	Pharma	> 10 years
2	Enterprise architect	Banking	9 years
3	Enterprise architect	Banking	> 10 years
4	Enterprise architect	Banking	> 10 years
5	Enterprise architecture consultant	Automotive	> 10 years
6	Enterprise architect	Insurance	7 years
7	Enterprise architect	Insurance	> 10 years
8	Enterprise architect	Automotive	6 years
9	Enterprise architect	Banking	> 10 years
10	Enterprise architect	Banking	> 10 years
11	Enterprise architect	Banking	> 10 years
12	Enterprise architecture consultant	Consulting	> 10 years

Table 5.17.: Overview about interviewed experts from industry for disparity metric evaluation

After a detailed presentation of the metric, its calculation rules as well as the exemplary results presented previously, four different questions have been used to assess the usability of the disparity metric for practitioners:

1. Do you confirm that in general an increase in diversity is related to an increase in the number of available capabilities?
2. Do you confirm that your budget for IT operations is limited or that currently IT costs should be reduced?

3. Are you currently using a similar approach to analyze which capabilities would get lost in case of standardization?
4. Would a disparity assessment provide you information relevant for decision making?

Regarding the first question, every single interviewee answered with *yes*. This indicates the general relevance of the disparity metric. Similarly, the second question has also been answered with *yes* by every interviewee. However, Interviewee 9 mentioned that the company he is working for just started to measure costs and therefore the numbers might increase in the long-run. Likewise, Interviewee 5 remarked that budget constrains only exist for IT operations. The novelty of the evaluated approach is confirmed by the fact that all interviewees stated that they do not use or are aware of any similar approach. Only variety is currently assessed by Interviewees 1, 2, 3, 4, 10 and 11 and balance is assessed only in the company Interviewee 4 is working for.

Regarding question four, answers are more diverse. In total, eight out of twelve interviewees agreed that the presented approach would be beneficial for them. For example, Interviewee 3 said that it is

> "*exactly what I need because we have to introduce a new NoSQL database to support new business demands but all metrics assessing application landscape diversity decrease. With disparity I would have an argument why it is no problem that other metrics decrease.*"

Likewise, Interviewee 5 explained his answer with the following statement:

> "*Considering and assessing disparity would fuel discussions about IT value in the company I am working for. Such discussions currently do not take place, only costs are discussed.*"

However, Interviewees 7, 8, 10 and 11, doubted that it is possible to define the required criteria for assessing disparity. For example, Interviewee 11 noted that

> "*it might be to hard to find consensus among all stakeholders regarding the distance function. Without consensus criteria might be considered arbitrary and metric results would lack acceptance.*"

Similarly, Interviewee 12 added:

> "*Disparity measurement according to the presented approach could provide an interesting perspective to enterprise architects if used as an internal tool. Achieving group-wide consensus might be too time-consuming.*"

Accordingly, it can be concluded that the disparity measurement based on the approach of Weitzman (1992) is not only novel in the context of EAM but is also expected to provide relevant information to support diversity management decisions in many cases. However, defining relevant characteristics is assumed to be time-consuming and consensus among all stakeholders might be difficult to achieve.

5.5. Integrated software support for diversity metric calculations

To enable enterprise architects to assess their application landscape's diversity based on the previously introduced indicators a software supporting required calculations is desirable. Therefore, such software support is described in this chapter which has been developed according to the design science paradigm.

To develop software support enabling enterprise architects to use the previously described complexity and diversity indicators an objective-centered entry point to the design science research method proposed by Peffers et al. (2007) is applied (see Figure 5.15). Therefore, the first step is to define the specific research problem and justify the value of the solution (see Chapter 5.5.1). The second step is to infer the objectives of a solution, i.e. requirements, from the problem definition and knowledge of what is possible and feasible (see Chapter 5.5.2). The third step is to create the artifact which in this case is, according to Hevner et al. (2004), an instantiation (see Chapter 5.5.3). The fourth step is to demonstrate the use of the developed software artifact to solve one or more instances of the problem (see Chapter 5.5.4). The fifth step is to observe and measure how well the artifact supports a solution to the problem (see Chapter 5.5.5). According to Peffers et al. (2007), this activity involves comparing the objectives of a solution to actual observed results from use of the artifact in the demonstration. The final step is to "communicate the problem and its importance, the artifact, its utility and novelty, the rigor of its design, and its effectiveness to researchers and other relevant audiences such as practicing professionals, when appropriate" (Peffers et al., 2007) which is achieved by this thesis.

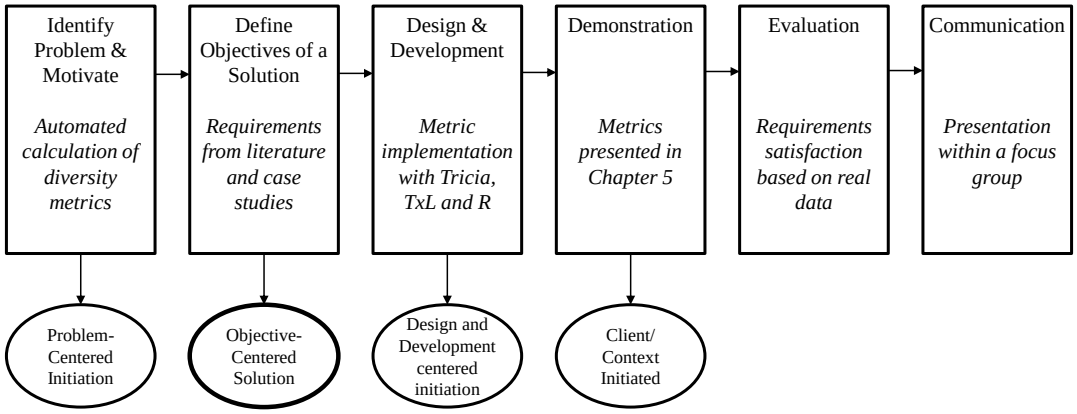


Figure 5.15.: The research approach to develop an integrated software solution for diversity metric calculations

5.5.1. Problem identification and motivation

Most organizations implementing EAM functions do not assess their application landscape's diversity, even if they are actively pursuing a decrease of diversity through standardization activities. Thus, many organizations invest in application landscape standardization programs without being able to evaluate whether their programs/measures lead to a decrease of diver-

sity. Without a formal diversity measure calculation they are not able to identify differences in standardization success among different architectural layers or business domains. The diversity metrics introduced in Chapter 5.4 provide such formal approach but rely on software performing the required calculations. Because some of these metrics have been developed only recently predefined implementations do not exist. Furthermore, the required capabilities necessary to define these metrics go beyond standard arithmetic or even SQL functions. A recent analysis of 13 EAM tools revealed that current tool support regarding the use of a Domain Specific Language (DSL) which might be helpful is also scarce (Hauder et al., 2013a). To use and visualize the results of metric calculations they need to be integrated with an EAM tool. The development and subsequent dissemination of a software supporting the calculation of diversity metrics that can be applied to arbitrary EA descriptions would enable researchers to assess the impact of different standardization methods across different companies. Further, such tool support would provide the means enabling practitioners to perform continued monitoring of their application landscape's diversity.

5.5.2. Objectives of the solution

The objective of the software solution to be developed is to allow enterprise architects to calculate diversity metrics based on an organization-specific EA model. The metrics to be supported include at least the metrics introduced in Chapter 5.4: the number of types, the balance of types (based on entropy and skewness formulas) and the disparity of types (based on the Weitzman metric). Furthermore, the solution needs to fulfill the following requirements:

R1 *Generic metric implementation.* The solution needs to implement metrics in a generic fashion which allows their application on arbitrary EA model elements because application landscape diversity is determined by the diversity of different EA elements (see Chapter 5.2).

R2 *Adequate performance.* Modeling application landscapes consisting of approx. 5,000 business applications and more than 20,000 information flows between them (as observed, e.g., for a German car manufacturing company) results in a significant amount of data. This data needs to be processed in order to calculate, e.g., the clustering metric introduced in Chapter 5.2 or the disparity metric introduced in Chapter 5.4.4. Because the complexity of the algorithms used to calculate these metrics is equal to $O(n^2)$, information processing needs to be efficient in order to provide timely results for enterprise architects.

R3 *Implementation reuse.* For many algorithms efficient implementations already exist. Therefore, the software to be developed should be able to exploit this fact rather then forcing the user to reimplement standard algorithms (e.g. the transitive closure for the classification metric introduced in Chapter 5.2) which might result in less efficient and error-prone metric implementations.

R4 *Type safety.* Matthes et al. (2013) argue why EAM tools should support emerging EA meta-models to avoid the so-called ivory tower syndrome. A software implementing diversity metrics which are calculated based on an emerging model is required to ensure type safety of metrics. When, for example, an attribute within the EA meta-model is renamed, all

metrics using the data referenced by this attribute need to be adjusted accordingly to ensure continuous operability.

R5 *Meta-information.* Previous research on the documentation of metrics in the context of EAM highlighted the necessity to document additional information for each metric. This includes, e.g., a description, measurement frequency, interpretation, consumer and owner role, and a target value (Matthes et al., 2012). Accordingly, the software to be developed needs to be able to allow such documentation for each diversity metric.

R6 *Change monitoring.* Observing EA metrics over time has been identified as an important requirement to provide actual decision support (Schneider et al., 2015c). Such time-series analysis requires a stable definition of it's corresponding metric. Therefore, any software supporting diversity design decisions has to monitor changes to metric definitions and provide adequate transparency.

R7 *Extensibility.* Ad-hoc information demands have been identified as a major challenge for enterprise architects (Hauder et al., 2013b). Therefore, the definition of complexity and diversity metrics has to be possible without creating too much implementation costs, e.g. for compilation, deployment or restart. Furthermore, a newly defined metric should be calculated with the latest data available.

5.5.3. Design and development

Given the requirements listed above, the next step towards the desired software solution is the development of a system design (Bruegge and Dutoit, 2010), i.e., a software architecture design. It decomposes the whole system into reasonable subsystems and regards the fact that EAM tools are already available on the market. During system design, the *Separation of Concerns* is a software engineering principle one should strictly follow (Dijkstra, 1982, pp. 60-66). By analyzing the requirements mentioned above, two major concerns can be distinguished. First, requirements R1 – R3 cover aspects related to the actual calculation of metrics. Second, requirements R4 – R7 cover aspects related to data and metric management which are typically addressed by EAM tools. Therefore, the proposed architecture design consists of two main components: one module for *calculation* and one module for *data and metric management*.

Implementing the *calculation* component with respect to requirements R1 – R3 is not a trivial task, particularly due to performance requirements. Therefore, already existing computing environments like R[1] or Matlab[2] might be used to realize the *calculation* component. The concrete tool choice depends on several aspects like availability of programming language knowledge, particular technical features or license costs. Therefore, the choice regarding the implementation of the calculation component might be organization-specific which implies that the *data and metric management* component needs to be loosely coupled with the *calculation* component in that sense that it makes no assumptions about its internals.

Therefore, a common interface for exchanging data between those two components has to be defined. The data exchange format needs to be compatible with a wide range of data sources as

[1] http://www.r-project.org/
[2] http://www.mathworks.com/products/matlab/

well as different computing environments. Especially in the case that two different calculation engines or data sources should be used in parallel such common protocol is needed. The most common data format for such purposes is the Comma-separated values (CSV)[3] format. This should be sufficient because there are no requirements known requiring a richer data exchange format. The design decision to allow different realizations of the calculation component has several implications on the *metric management* component. First, metric calculation rules need to be developed in file formats readable by the chosen *calculation* component. Therefore, the *data and metric management* component needs to be able to store such file formats.

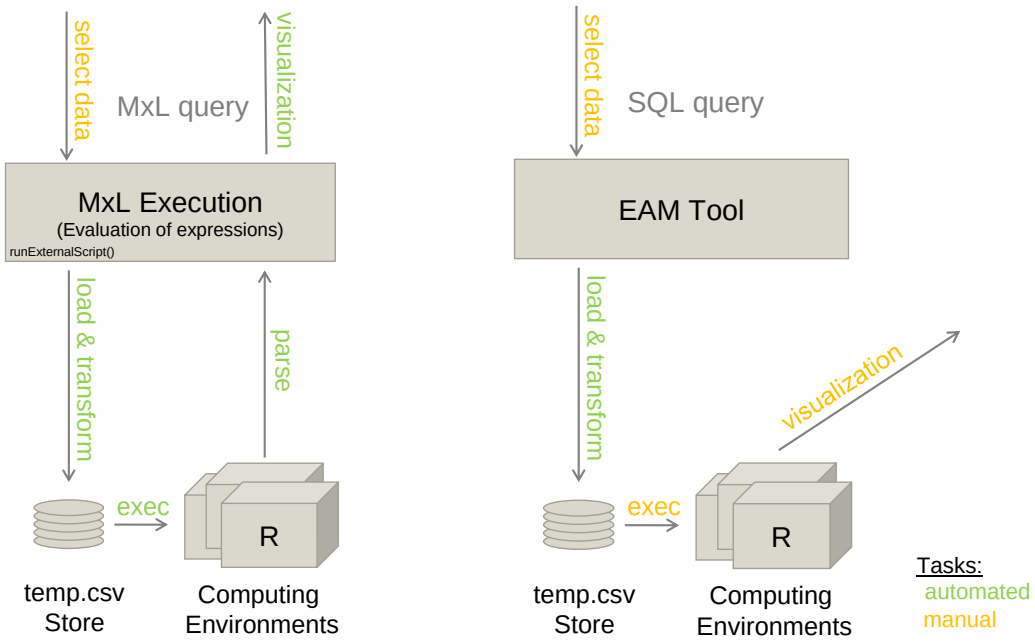
Implementing the *data and metric management* component from scratch might also not be expedient because different EA tools already exist on the market. Furthermore, most companies might not be willing to change their EA tool just to perform metric calculations. Because the scope of the desired software solution covers only metric management and calculations in the context of EA modeling, it is assumed that such *data management* component is already in place. This component can be a simple spreadsheet or database application storing all relevant EA information in one case or a comprehensive EAM tool offering a wide range of features in another case. Because the technical features and integration possibilities vary largely among available EA tools (Matthes et al., 2008), two different scenarios need to be distinguished. First, if a simple data management component (e.g. Microsoft Excel) is used to store EA information, data selection needs to be done manually to provide the necessary CSV-Interface to be used by the *calculation* component. In this scenario, the calculation component is also responsible for visualizing and rendering calculation results. Second, if an advanced EA tool is used which is able to delegate the calculation of metrics to the *calculation* component, this tool is also responsible for processing the calculation results. In such case, the user gets a single integrated interface to interact with EA data.

To best of our knowledge, no EA tool is able to simply integrate computation environments like R or Matlab. However, to build a solution satisfying the requirements stated in Chapter 5.5.2, an existing tool satisfying requirements R4 – R7 has to be extended. This extension covers mainly a delegation mechanism for performant metric calculations. Therefore, the design of the prototypical implementation of the desired software solution is based on two existing tools which will be combined to fulfill the requirements listed in the previous chapter: Hybrid Wikis (Matthes et al., 2011) and R. R is a language and environment for statistical computing and graphics. Therefore, it will be used to address requirements R1 – R3. Hybrid Wikis are an Enterprise 2.0 tool which has been successfully used to support EAM initiatives (Matthes et al., 2013). Hybrid Wikis in combination with the Model-based Expression Language (MxL) (see Reschenhofer et al., 2014) are used to address requirements R4 – R7.

Accordingly, the architecture of the prototypical implementation consists of two main components:

R environment The R environment is responsible for executing metric calculations. The definition of a metric needs to be passed as an R script. External libraries necessary to calculate metrics need to be installed by the administrator manually or via respective instructions within the particular script. The R language provides a rich set of predefined functions allowing a fast implementation of metrics.

[3] http://tools.ietf.org/html/rfc4180



(a) HybridWiki scenario

(b) Generic EA tool scenario

Figure 5.16.: Two scenarios for the structure and control flow of the intended software solution

TxL and Tricia The TxL module of the Hybrid Wiki implementation *Tricia* is responsible for selecting and projecting the data required for metric calculation. In addition, it is used to render and cache the metric results for the user. It is based on MxL (Reschenhofer et al., 2014).

Figure 5.16 shows the intended flow of actions when a metric should be calculated for each of the two possible scenarios. In both scenarios metric calculation rules are implemented in a programming language interpretable for the chosen computation environment.

The HybridWiki scenario depicted in Figure 5.16a starts with the manual task of data selection. In this scenario, the user selects the data he or she wants to use for metric calculation by defining a MxL query. Here, the MxL language can also be used to define the data format expected by any particular metric (for details refer to Chapter 5.5.4). Consequently, the type checker of MxL is able to validate each data selection before actually executing it to prevent erroneous data selection queries. Based on the data selection query, the MxL component is responsible for loading the required data from the underlying database and store it temporally in CSV format. In this prototypical implementation, the MxL component needs to be aware of the R installation in order to call it. Therefore, a small extension to the existing MxL component has been implemented: the *runExternalScript* method. The signature of this method is as follows:

```
1  runExternalScript(pathToComputationEnvironment:String,
2   scriptToExecute:String,
3   fileInput :Set<Set<Object>>,
4   textInput:Set<String>)
5
```

Listing 5.1: The runExternalScript method for executing metric calculations

Accordingly, the new method takes the local installation path to the chosen computation environment, e.g. R, as first parameter. As a second parameter, the actual script, i.e. the calculation rule of a metric, needs to be given. This allows the storage of metric calculation rules as HybridWiki elements. Not only can HybridWiki implementations enforce access rights on metric definitions but also track changes. Furthermore, the experienced user can define metric calculation rules directly within the HybridWiki environment without opening a shell or editor specific a particular computation environment. The third parameter is a set of sets containing objects/entities stored within HybridWikis, i.e. the data. Finally, the fourth parameter is a set of additional strings which should be passed to the calculation environment as additional arguments. For convenience, specialized MxL methods for available calculation environments can be added during runtime. For the R environment used in this prototype the following method has been defined:

```
1  runRScript(scriptToExecute:String, dataInput:Set<Set<Object>>,
2   textInput:Set<String>) {
3     runExternalScript("concreteRpath", scriptToExecute, dataInput, textInput);
4  }
5
```

Listing 5.2: A convenient MxL expression for executing R scripts

When such method needs to be evaluated, the MxL component will first load the objects specified by ordinary MxL operators within the third argument. Due to already available MxL functionality, selection and projection operations can be performed on the underlying EA data. After loading the specified data, the *runExternalScript* method transforms this data and stores it temporally as a CSV file. Then, the script provided as second parameter will also be stored temporally. Finally, it will execute a command within the operating system's command prompt based on the given path (first parameter), the actual script to be executed (path to the created file) and additional parameters (path to the created CSV files and text parameters). To allow also standalone operation of the calculation component it is configured to write the result of the script execution back to the command prompt. All values appearing here will be parsed automatically by the MxL component and mapped to a simple table. These values are than accessible for the user and can be used for additional calculations or visualizations within MxL expressions. A detailed example including screenshots is presented in the subsequent section.

Because not every company uses HybridWikis to store EA related information, this prototype is build to support also the scenario in which an arbitrary EAM tool is used (see Figure 5.16b). Although the main control flow is similar, in this scenario the degree of automation and result processing is different. First, data selection routines might probably not be stored within the EAM tool. Therefore, it might be necessary to store Structured Query Language (SQL) statements or similar queries somewhere else and execute them when necessary. The selected

data needs then be transformed to be stored as CSV file because this is the generic data format understood by most calculation environments. After script execution the results might not be sent back to the EAM tool because of missing interfaces. Therefore, the built-in functionality of calculation environments has to be used. All common platforms provide numerical and graphical result representations which allows also this stand-alone scenario. The next section demonstrates how previously described diversity metrics can be implemented with R and their integration in the HybridWiki implementation Tricia.

5.5.4. Prototype demonstration

To demonstrate the feasibility of diversity metric calculation using the previously introduced software solution based on R and the HybridWiki implementation Tricia a concrete use case will be described. The implementation of diversity metrics presented in Chapters 5.2 and 5.4 is performed by using the R environment and publicly available libraries. Together, all metric implementations are bundled within an R library named *diversity* for easier exchange and installation. This library includes the following methods for which the library name is later on used as prefix:

- variety (counts the number of concepts)
- balance.nominal (calculates Shannon entropy of concepts and individuals)
- balance.ordinal (calculates skewness for ordered concepts and individuals)
- disparity (calculates the Weitzman metric for concepts)
- disparity.minLoss (returns the set of elements minimizing disparity loss when removed)
- dendrogram (renders a dendrogram of concepts based on euclidean distance)

The first implemented metric covers the variety aspect of diversity. As depicted in Listing 5.3, it counts the number of given types. It makes no assumptions about these types so it can be used to count the number of different operating systems in use exactly like different database management systems.

```
1  diversity.variety <- function(observations) {
2    length(unique(observations))
3  }
4
```

Listing 5.3: R implementation of the variety indicator

The next indicator which has been implemented is the balance indicator based on the Shannon entropy (see Shannon, 1948). The implementation depicted in Listing 5.4 makes use of the *entropy package*[4] and is therefore only three lines long. This demonstrates the reuse of existing implementations (see requirement R3).

[4] http://cran.r-project.org/web/packages/entropy/entropy.pdf

```
diversity.balance.nominal <- function(observations) {
  entropy::entropy.empirical(observations)
}
```

Listing 5.4: R implementation of the nominal balance metric

To complement balance calculations based on nominal data, the ordinal-based metric has been implemented according to Listing 5.5. Thereby, an inner function is defined first, to calculate cummalitve absolute frequencies ($F_X(x_i)$). After summing up all cumulative absolute frequencies, the intermediate result is divided by the total number of instances because relative frequencies need to be used. Finally, the division by k leads to the normalized result of this indicator.

```
diversity.balance.skewness <- function(data, normalize=TRUE) {
  # Inner function to calculate cumulative frequencies
  cumulativeFrequency <- function(data, i) {
    sum(data[1:i])
  }

  # Number of variable states minus 1
  k <- length(data)-1

  # Result variable
  r <- 0

  # Sum of all cumulative frequencies
  for(i in 1:k) {
    r <- r + cumulativeFrequency(data, i)
  }

  # Formula according to Ingo Klein
  # Division by sum(data) is necessary because relative frequencies are required and
  # above absolute frequencies have been used
  r <- -k+(2*r/sum(data))

  # Normalization if desired
  if(normalize) {
    r <- r/k
  }

  return(r)
}
```

Listing 5.5: R implementation of the ordinal balance indicator

The fourth metric implementation addresses the disparity aspect of diversity. It calculates the metric proposed by Weitzman (1992) based on the input *data*. In addition, two parameters can be set to overwrite their default value. First, it is assumed that the *data* has no type names included in the first column. If so, the first column of the *data* table has to be removed because these names will not be used during distance calculations. If the *data* is provided with such type names, the optional parameter *removeFirstColumn* can be set to TRUE. By default, distances

will be calculated using the euclidean distance. If another distance function should be used, the second optional parameter *dist_method* can be set to one of these values: `maximum, manhattan, canberra, binary` or `minkowski`. For example, the `binary` method can be chosen, if disparity characteristics are only modeled in a yes/no fashion. What will happen is that the Jaccard distance (see Jaccard (1912)) will be calculated instead of the euclidean distance.

First, depending on the value of the *removeFirstColumn* parameter, the first column of the provided data set will be removed (lines 3–5). Within the next step (line 8), the distance matrix is calculated based on the value of the *dist_method* parameter. For further processing, the result is transformed to the matrix data type. The next three lines (9–11) only initialize variables used during the upcoming calculation. The `originVector` variable includes all columns of the distance matrix except the first one. The reason is that the first column, i.e. the first element of the input data, is directly assigned to the `resultVector` variable. Instead of the first column, the first row could also be chosen due to the fact that a distance matrix is always symmetric. The `weitzman` variable will hold the intermediate results for the metric and is returned at the end. Next (line 14), within the distance matrix the values on the diagonal representing the distance from each element to it self (which is usually 0) is set to infinity. This eases the search for the minimum distance within the distance matrix for pair-wise elements (x, y) for which $x \neq y$ holds.

In line 17, the actual metric calculation begins. For each type to be processed, i.e. each column of the distance matrix, the type (column) including the smallest distance value to the result set is identified. This is done by the `min()` function which uses only the `resultVector` part of the distance function. In the first iteration, this would be only the first type we already selected. In line 19, the index of that type is selected based on the identified distance minimum. Then, the identified minimum distance value needs to be added to the intermediate result. Therefore, the `weitzman` variable is incremented accordingly (line 21). The identified type will then be added to the `resultVector` variable (line 22). Finally, all row and column values within the distance matrix belonging to the types already considered, i.e. included in the `resultVector` variable, will be set to infinity to neglect them during the next iteration because we are not interested in distance values among already processed types. The last step of the disparity method is to return the accumulated value of least distances among given types stored in the `weitzman` variable.

The benefit of the metric proposed by Weitzman (1992) is that it can be used to identify those types which minimize the lost disparity if removed. Therefore, an additional implementation is provided named *disparity.minLoss* which is depicted in Listing 5.7. After initializing a vector of length equal to the number of types included in the input `data` (line 4), a separate vector is initialized to store the names of the given types (line 7) so that the plain data can be used for further processing (line 10). Then, the disparity metric is calculated for all data subsets for which exactly one type has been removed (lines 13 – 16). Thereby, it makes use of the previously introduced disparity function (see Listing 5.6) and stores results in the `weitzmanResults` variable. Finally, the *disparity.minLoss* function returns the names of the types for which the remaining disparity is maximized (line 21).

```
diversity.disparity <- function(data, removeFirstColumn=FALSE, dist_method="euclidean") {
  # Delete the first column of input data if it contains names
  if(removeFirstColumn) {
    data <- data[,2:ncol(data)]
  }

  # Load data and initialize variables
  distance_matrix <- as.matrix(dist(data, method=dist_method, diag=T))
  originVector <- c(2:ncol(distance_matrix))
  resultVector <- c(1)
  weitzman <- 0

  # Normalize table
  for (i in 1:ncol(distance_matrix)) { distance_matrix[i,i] <- Inf }

  # Calculate metric
  for (i in 2:ncol(distance_matrix)) {
    minVal <- min(distance_matrix[,resultVector])
    minInd <- which(distance_matrix[,resultVector] == minVal)%%nrow(distance_matrix)
    if (minInd[1] == 0) minInd <- nrow(distance_matrix) else minInd[1]
    weitzman <- weitzman + minVal
    resultVector <- c(resultVector, minInd[1])
    distance_matrix[minInd[1],resultVector] <- Inf
    distance_matrix[resultVector,minInd[1]] <- Inf
  }

  #Return result
  weitzman
}

```

Listing 5.6: R implementation of the disparity metric

```
diversity.disparity.minLoss <- function(daten) {
  # Initialize vector to store Weitzman results
  weitzmanResults <- rep(c(NA),each=nrow(daten))

  # Store type names in separate vector
  names(weitzmanResults) <- daten[,1]

  # Store data without type names in separate matrix
  data <- daten[,2:ncol(daten)]

  # Calculate Weitzman diversity for all combinations where exactly one type is omitted
  for(i in 0:(nrow(data)-2)) {
    subdaten <- data[c(0:i,(i+2):nrow(data)),]
    weitzmanResults[i+1] <- diversity.disparity(subdaten)
  }

  # Calculate Weitzman diversity also for the last combination where the last type is omitted
  weitzmanResults[nrow(data)] <- diversity.disparity(data[c(0:i,(i+2):nrow(data)),])

  # Print names of types which can be removed in order to maximize the remaining Weitzman diversity
  names(which(weitzmanResults==max(weitzmanResults)))
}

```

Listing 5.7: R implementation for decision support regarding type removal

To support also the stand-alone scenario (see Chapter 5.5.3) the diversity library also includes a function to plot a dendrogram (see Listing 5.8). It is a tree-like diagram frequently used to illustrate the arrangement of clusters, e.g. based on the distance calculated by the disparity metric. Therefore, it takes the input `data` as a first parameter and a boolean value `removeFirstColumn` as second parameter indicating whether the first column of the provided data needs to be removed because it includes the names of the types (lines 1 – 4). Similar to the disparity metric, the distance matrix is calculated for the given input data using the euclidean method (line 6). By using the `hclust` function, a hierarchical cluster analysis is performed using the calculated distances. Initially, each object is assigned to its own cluster and then the algorithm proceeds iteratively, at each stage joining the two most similar clusters, continuing until there is just a single cluster[5]. By calling the `plot` function on the result of the `hclust` function the actual image showing the dendrogram is created (line 8).

```
1  diversity.dendrogram <- function(data, removeFirstColumn=FALSE) {
2    if(removeFirstColumn) {
3      data <- data[,2:ncol(data)]
4    }
5
6    distance_matrix <- dist(data, method="euclidean", diag=T)
7    hierarchical_cluster_analysis <- hclust(distance_matrix)
8    plot(hierarchical_cluster_analysis, hang=-1, labels=daten[[1]])
9  }
10
```

Listing 5.8: R implementation for rendering a dendrogram based on the disparity metric

5.5.5. Technical evaluation and conclusion

In this section, it is evaluated in how far the requirements presented in Chapter 5.5.2 are reached by the software solution presented in the previous section. Therefore, concrete instances of the diversity metrics introduced before will be implemented and applied to application landscape data which has already been used in Chapter 5.3.

Preparation

To set up the prototypical implementation, it is necessary to set up an instance of the HybridWiki implementation Tricia including a relational database management system on a server. Then, the R environment needs to be installed on the same machine. Because the *diversity* library relies on the *entropy* library, this library needs to be installed as well. Finally, Tricia needs to be told the installation path of R in order to be able to start it. The easiest way of doing this is to create a MxL function named `runRscript` for this purpose during runtime. As depicted in Listing 5.9, it uses the built in function `runExternalScript` (see Listing 5.1) and statically passes the R installation path as first parameter. The `runRscript` function might, for example,

[5] https://stat.ethz.ch/R-manual/R-patched/library/stats/html/hclust.html

look like Listing 5.9. The second, third and fourth parameter of the `runExternalScript` method are simply passed as parameters of the `runRscript` function.

```
1  (script:String, data:Sequence<Sequence<Object>>, params:Sequence<String>) =>
2    runExternalScript("C:\Programs\R\R-3.1.2\bin\x64\Rscript.exe", script, data, params)
3
```

Listing 5.9: MxL function for conveniently executing R scripts

Variety metrics

To calculate the variety of operating systems as well as database management systems two different metrics should be defined by adhering to the following structured approach:

1. Store R script for variety calculations in Tricia as HybridWiki instance of type *Script* and add an attribute named *name* with the value *variety*.
2. Develop MxL functions for the R script defining its interfaces
3. Create a new wiki page and instantiate these functions by selecting concrete data

While the first step is straightforward and needs no further explanations, the second step is important to ensure type safety, for example, for operating systems and database management systems. Listing 5.10 shows such implementation to be used for calculations using operating systems data. Within `varietyOfOperatingSystems` definition, the type of the two parameters can be further constrained. To be type safe with operating systems, the type of the first parameter named `data` needs to be set to `Sequence<'Operating system'>` because Sequence is the data type for sets in MxL and *Operating system* is the HybridWiki type used to identify operating systems. By putting this parameter into square brackets when calling the `runRScript` function it is guaranteed that only one data set is used as required by the variety function (see Listing 5.9) although the `runRScript` function generally allows multiple data sets. The second parameter can be omitted because the variety metric does not need any further parameters, i.e. an empty sequence can be used.

```
1  (data:Sequence<'Operating system'>) =>
2    runRScript(find('Script').single(name="variety"), [data], [])
3
```

Listing 5.10: MxL function to describe the interface of variety calculations for operating systems

Up to now, the calculation rule for the variety metric has been uploaded to the HybridWiki implementation Tricia and respective MxL function definitions, i.e. interfaces, have been developed to ensure type safety when calling this metric. Therefore, the only step missing is the concrete instantiation of this metric. This is usually done on a new wiki page on which concrete metric results should be displayed. Listing 5.11 shows such implementation for the variety of *operating systems*. In addition, Listing 5.12 shows another implementation in which variety is calculated only for *Operating systems* used by *Business applications* belonging to the *Card management*

domain. After selecting the desired data, it needs to be transformed to comply with the CSV format by calling the `asCSV()` function.

```
1  let data = find('Operating system').asCSV() in
2  varietyOperatingSystems(data)
3
```

Listing 5.11: MxL function for calculating the variety of operating systems

As one can see easily, it is possible to perform arbitrary data selections by using the MxL language inside Tricia and calculate the variety metric accordingly. Because the `varietyOperatingSystems` metric has been defined as an interface to the generic variety metric, type safety can be guaranteed.

```
1  let domain = find('Domain').where(name="Card management") in
2  let applications = domains.selectMany(get 'Business Application' whereis belongsTo) in
3  let data = applications.selectMany(get 'Operating system' whereis uses).asCSV() in
4  varietyOperatingSystems(data)
5
```

Listing 5.12: MxL function for calculating the variety of operating systems (advanced)

Balance metrics

Similar to the variety metric described in the previous section, the R implementation of the balance metric (see Listing 5.4) is stored in Tricia as HybridWiki instance of type *Script*. Its *name* attribute is set to *balance* to enable an easy access to the metric implementation via MxL. Then, for the convenient usage, an MxL function acting as an interface to the metric implementation is defined. An exemplary MxL function for calculating balance of *Operating systems* named `balanceOperatingSystems` is shown in Listing 5.13. It takes only a sequence of *Operating systems* as a parameter. Thereby, it is guaranteed, that exactly one data set is sent to the `runRScript` function although in general it allows multiple data set.

```
1  (data:Sequence<'Operating system'>) =>
2    runRScript(find('Script').single(name="balance"), [data], [])
3
```

Listing 5.13: MxL function to describe the interface of balance calculations for operating systems

Again, the third parameter of the `runRScript()` function can be omitted because the balance implementation does not require any additional parameters. To actually use this metric, the third step is to create a new Wiki page on which metric results should be shown. If, for example, the balance of Operating systems within a specific domain has to be calculated, the respective MxL call might look like Listing 5.14. Similarly, the ordinal balance metric could be implemented and used.

```
1  let domain = find('Domain').where(name="Card management") in
2  let applications = domains.selectMany(get 'Business Application' whereis belongsTo) in
3  let data = applications.selectMany(get 'Operating system' whereis uses).asCSV() in
4  balanceOperatingSystems(data)
5
```

Listing 5.14: MxL function for calculating the balance of operating systems

Disparity metric

The first step in implementing the disparity metric in Tricia is similar to the two metrics described in the previous sections: store the R implementation as a new HybridWiki instance of type *Script* and give it a unique *name* attribute value, e.g. *disparity*. In contrast to the variety and balance functions, the disparity function is able to parse additional parameters. According to its definition (see Listing 5.6), the first additional parameter is a flag indicating if the provided data still contains names which need to be removed. The second additional parameter indicates the type of distance function to be used. Therefore, the MxL function needs to account for this fact. Because MxL is used to select appropriate data, the first additional parameter is not needed here. However, the second parameter is needed to allow the user to specify the kind of distance function to be used. Therefore, the `disparityOperatingSystems` function not only takes a typed parameter for data selection but also takes a parameter for distance method specification. The `?` behind the type specification of this parameter indicates that this is an optional parameter. Therefore, in line 2 the function needs to check whether it has been supplied or not. If a value has been supplied, this value is passed to the `runRScript` function. Otherwise, an empty sequence is passed.

```
1  (data:Sequence<'Operating system'>, method:String?) =>
2    let param = if method <> null then ["method='' + method)] else [] in
3    runRScript(find('Script').single(name="disparity"), [data], param)
4
```

Listing 5.15: MxL function describing the interface of disparity calculations for operating systems

In comparison to the other two metrics, the disparity metric expects a more comprehensive data format. To calculate pair-wise distances between elements, the desired characteristics and respective manifestations for each instance have to be selected. Listing 5.16 shows a possible instantiation for *Operating systems*. The selection of the desired operating system characteristics occurs within the `asCSV()` function. Here, five different attributes are selected to calculate pair-wise distances.

```
1  let domain = find('Domain').where(name="Card management") in
2  let applications = domains.selectMany(get 'Business Application' whereis belongsTo) in
3  let data = applications.selectMany(get 'Operating system' whereis uses).asCSV([Server, Client,
    Embedded, Real-time, GUI]) in
4  disparityOperatingSystems(data)
5
```

Listing 5.16: MxL function for calculating the disparity of operating systems

Requirements satisfaction

In the previous sections, it has been evaluated in how far the identified metrics regarding the three aspects of diversity can be implemented within the HybridWiki implementation Tricia. By using Tricia's expression language MxL to select required data stored within Tricia and by calling the `runExternalScript()` function which delegates calculations to an R environment all metrics can be realized in Tricia.

Regarding the requirement for generic metric implementation (R1), it can be concluded that the designed solution fulfills this requirement. Metric implementations are done by using the R language without referring to any types or data within a concrete EA model. Then, MxL is used to define desired interfaces for concrete metrics. Because data selection is also done with MxL arbitrary EA models are supported.

Regarding the requirement for adequate performance of metric calculations (R2), we can state that for a typical size of input data metric calculations are performed timely. Variety and balance metrics are in $O(n)$ and the number of considered EA elements typically does not exceed 10,000. Therefore, a common desktop computer can perform these calculations in less than 1 second. Although the disparity metric is in $O(n^2)$ input data is usually not very large. Usually, a company might not use more than 100 database systems or operating systems. In the four cases described in Chapter 5.3.2, not more than ten different types of operating systems or database systems have been observed. In addition, many R functions and libraries are implemented in low-level programming languages like C and Fortran which minimizes computation overhead.

Regarding the requirement for implementation reuse (R3), we can state that by using an existing open source statistical computing environment like R this requirement is fulfilled. The reuse of existing implementation has been exploited, for example, during development of the balance metric (see Listing 5.4). There, the existing library *entropy* has been used without any further implementation efforts.

Regarding the requirement for type safety (R4), we can state that this requirement is fulfilled by the proposed software solution. Type safety is guaranteed by using MxL which provides this feature out of the box. When implementing a new metric, the user just has to define a new MxL custom function which uses the desired types and maps them to the untyped R implementation. In addition, if the underlying EA meta-model changes, types used within metric definitions are changed automatically.

Regarding the requirement for additional information (R5), we can state that the proposed software solution fulfills it. The requirement is addressed in so far that each metric within the HybridWiki implementation Tricia can be enriched with additional structured and unstructured information. As described before, the R code of each metric is stored as a HybridWiki instance. Therefore, it can be documented at runtime like any other HybridWiki instance.

Regarding the requirement for change monitoring (R6), we can state that it is fulfilled by the same means like requirement R5. Tricia provides a change history for every HybridWiki instance and therefore also for metrics including the generic implementation as well as each concrete instantiation. Therefore, each change made to metric is recognized and previous versions can be restored.

Regarding the requirement for extensibility (R7), we can state that HybridWiki fulfills it by nature. As demonstrated above, each metric has been added to the system during runtime. Therefore, it is possible to implement new metrics also at runtime without a need for a server shutdown or re-deployment. Ad-hoc information demands can therefore be satisfied.

CHAPTER 6

Using system dynamics to support application landscape design decisions

"Most important, and most difficult to learn, systems thinking requires understanding that all models are wrong".

John D. Sterman, *Jay Wright Forrester Prize Lecture, 2002*

In previous chapters, suitable indicators able to quantify diversity of application landscapes have been presented. They assist enterprise architects in measuring diversity to analyze the current application landscape, define target values for future states of the application landscape and monitor the impact of design decisions over time. However, these indicators fall short in explaining why diversity increases or decreases over time because only structural aspects are regarded and it is assumed that a single designer or architect is in place who is able to steer diversity. As already outlined in Chapter 2.1.2, enterprises should be regarded as complex systems. Therefore, they are not only subject to structural complexity which can be assessed by previously presented indicators but also inhabit dynamic complexity. This includes, for example, phenomena like emergence (Morel and Ramanujam, 1999; Goldstein, 1999), non-linear behavior (Poincaré, 2003) and path-dependence (David, 1985) and they are often made up by autonomous agents without any central control (Mitchell, 2011). Therefore, beside the ability to quantify diversity, understanding causal dependencies within an organization influencing diversity becomes vital for enterprise architects. Knowledge about how actors, such as business unit managers, enterprise architects and software developers, influence each other's decisions leads to improved alignment and coordination of actors' decisions. Hence, when considering principles guiding the architectural evolution, we need to account for both explicit principles defined by enterprise architects (Richardson et al., 1990; Greefhorst and Proper, 2011) as well as implicit principles emerging within the complex system. To cover implicit principles and model causal

relationships influencing an EA, several authors already proposed to extend EA descriptions with System Dynamics (SD) models (Kaisler et al., 2005; Rashid et al., 2009; Golnam et al., 2010) and social aspects (Molnar and Korhonen, 2014).

To get one step further towards a method enabling enterprise architects to develop SD models (see Forrester, 1961; Sterman, 2000), this chapter outlines based on Schneider et al. (2015a) the state-of-the-art in SD modeling within the context of EAM by performing a structured literature review to motivate the approach and describe the problem. Based on existing work on SD modeling, design guidelines for a method enabling enterprise architects to develop SD models are developed. To demonstrate the guidelines' applicability a concrete Causal Loop Diagram (CLD) describing diversity management activities occurring within companies based on input gathered from literature is derived. This prototypical CLD is then evaluated by a series of expert interviews. Based on the analyzed interview results additional design guidelines accounting for EAM specifics are derived.

6.1. System dynamics foundations

"Without modeling, we might think we are learning to think holistically when we are actually learning to jump to conclusions" (Senge, 1990, p. 183). Instead of relying on gut feeling, a well-crafted model can close the feedback loop by which humans learn by showing the implications of their assumptions (see Figure 6.1). To model also dynamic behavior of systems, System Dynamics (SD) can serve as powerful tool by integrating both mathematical and methodological aspects. Initially proposed by Forrester (1961), SD is an approach to understanding the behavior of complex systems over time. Thereby, two different types of models are commonly used: Causal Loop Diagrams (CLDs) and Stock-and-Flow Diagrams (SFDs). Thereby, CLDs provide a macroscopic view on causalities within a system whereby SFDs provide a more detailed and quantitative view on a system allowing also for simulations (Agyapong-Kodua et al., 2012).

6.1.1. Learning in and about complex dynamic systems

The discipline of systems thinking tries to understand a system by analyzing dependencies and interactions between its elements. Within that discipline, different schools of system thinking exist (Richardson, 1999; Lane, 1993). For example, some authors argue for qualitative methods (e.g. Checkland, 1985) while others emphasize formal modeling (e.g. Forrester, 1961). "The challenge facing all is how to move from generalizations about accelerating learning and systems thinking to tools and processes that help us understand complexity, design better operating policies, and guide organization- and society-wide learning" (Sterman, 1994). Thereby, all learning depends on feedback (Forrester, 1961). As visualized in Figure 6.1, decisions influencing the real world are based on feedback from the real world. This basic feedback loop can also be observed in management literature. For example, the Plan-Do-Check-Act cycle (Shewhart and Deming, 1939) describes exactly this kind of feedback loop. Thereby, "decision makers compare quantitative and qualitative information about the state of the real world to various goals, perceive discrepancies between desired and actual states, and take actions that (they believe will) cause the real world to move toward the desired state" (Sterman, 1994). However, decisions are

also the result of applying decision rules or policies to information (Forrester, 1961) which are themselves conditioned by institutional structures, organizational strategies and cultural norms. These in turn are governed by the mental models of the real world hold by the decision maker. Argyris (1985) calls this kind of learning single-loop learning because the mental models of the decision maker stay unchanged. Therefore, it is a process whereby the decision maker learns to reach his or her current goals in the context of his or her existing mental models. Based on the works of Axelrod (2015); Bower and Morrow (1990); Cheng and Holyoak (1985); Schank and Abelson (2013), mental models can be defined as follows.

Definition: Mental model
Mental models are collections of routines, scripts, or schemata for selecting possible actions, cognitive maps of a domain, typologies for categorizing experience, pointers from instances of a phenomenon to analogous instances, logical structures for the interpretation of language, or attributions about individuals we encounter in daily life.

(Sterman, 1994)

In contrast to single loop learning, double loop learning occurs if information feedback is used to change mental models (Argyris, 1985) which is illustrated by Sterman (2000) in Figure 6.2. Hereby, the same information, filtered and processed through different decision rules might yield a different decision. Systems thinking targets at enabling this kind of learning for decision makers.

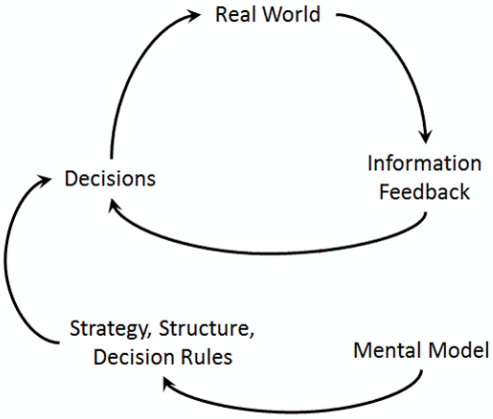


Figure 6.1.: Single loop learning according to Sterman (2000)

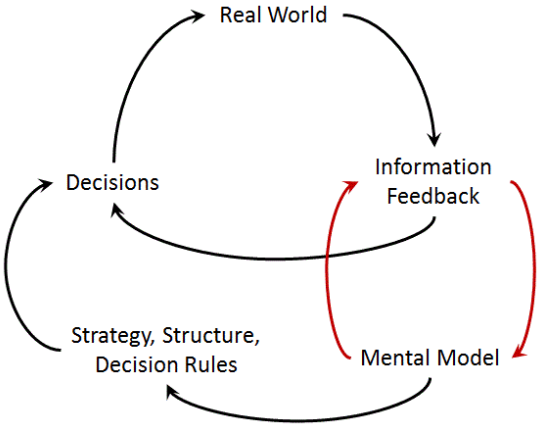


Figure 6.2.: Double loop learning according to Sterman (2000)

Forrester (1961) identified several barriers to learning via single loop learning. For example, feedback loops may reflect both anticipated and unanticipated side effects of the decision maker's actions, there may be positive as well as negative feedback loops and these loops might contain nonlinearities. In addition, time delays between taking a decision or action and observing the effects are common and problematic, information about the system under investigation is missing and controlled experiments are not feasible. One means to overcome such barriers to learning

is using virtual worlds for experimentation (Sterman, 1994). Therefore, at least two different kinds of modeling approaches exist which are briefly introduced in the following chapters.

6.1.2. Causal Loop Diagrams

As Sterman (2000, p. 137) points out, "Causal Loop Diagrams (CLDs) are an important tool for representing the feedback structure of systems. Long used in academic work, and increasingly common in business, CLDs are excellent for quickly capturing your hypotheses about the causes of dynamics, eliciting and capturing the mental models of individuals or teams and communicating the important feedbacks you believe are responsible for a problem". CLDs consist of two main types of elements: variables and arrows denoting causal influences among these variables. Usually, important feedback loops are also identified in diagrams. Figure 6.3 shows a simple example for a CLD. Therein, the variable *Birth Rate* is determined by both the *Population* and the *Fractional Birth Rate*. Each causal link is assigned a polarity, either positive (+) or negative (–) to indicate how the dependent variable changes when the independent variable changes. A plus sign indicats a positive relation between two variables implying that they change in the same direction. A minus sign indicates a negative relation implying that they change in the opposite direction. Consequently, a closed loop of such relations has either a balancing or a reinforcing effect on involved system variables. The type of such loop is typically indicated by respective symbols in their center. In addition, if cause and effect are subject to a significant delay in time, two parallel lines perpendicular to the respective causal link can be used to indicate such time delay.

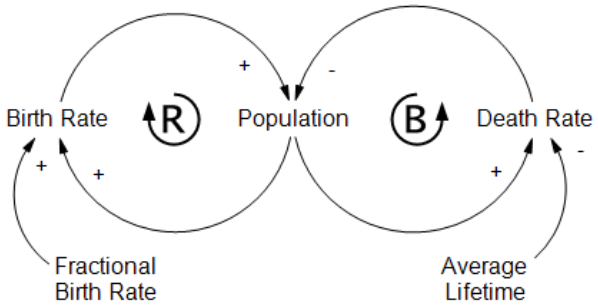


Figure 6.3.: Causal loop diagram notation according to Sterman (2000)

In the given example, a reinforcing cycle exists between *Birth Rate* and *Population*. If the population increases, the number of births will also increase. If the number of births increases, the population will increase. However, one cannot determine if, for example, the birth rate will increase at a given point in time, because it depends on more than one cause, i.e. both the fractional birth rate and the size of the population. A huge collection of CLDs providing additional examples from different disciplines has been developed by Bossel (2004b) and should be consulted by the interested reader.

Unfortunately, CLDs do not distinguish between stocks accumulating resources over time and flows altering such stocks. In the given example, *Population* is a stock. In contrast, the *Birth Rate* is a flow which can only add new individuals to the population. It can never decrease

it. Therefore, a second type of SD diagram can be used which is introduced in the following chapter: Stock-and-Flow Diagrams.

6.1.3. Stock-and-Flow Diagrams

According to Sterman (2000), CLDs are especially useful to capture mental models and to communicate the results of a completed modeling effort. However, to overcome their limitation in capturing the stock and flow structure of a system, Stock-and-Flow Diagrams (SFDs) can be used to complement CLDs. Thereby, stocks give systems inertia and provide them with memory. They also create delays by accumulating the difference between inflows to a process and its outflows. Within SFDs, different modeling elements can be used. For example, stocks are represented by rectangles and flows by pipes pointing into or out of a stock. Valves are used to control flows. To model sources or sinks of flows outside the boundary of the model, the cloud symbol is used. According to Mass (1977), stocks are critical in generating the dynamics of systems. For example, they characterize the state of the system and provide the basis for actions and decouple rates of flow and create disequilibrium dynamics. An exemplary SFDs is shown in Figure 6.4.

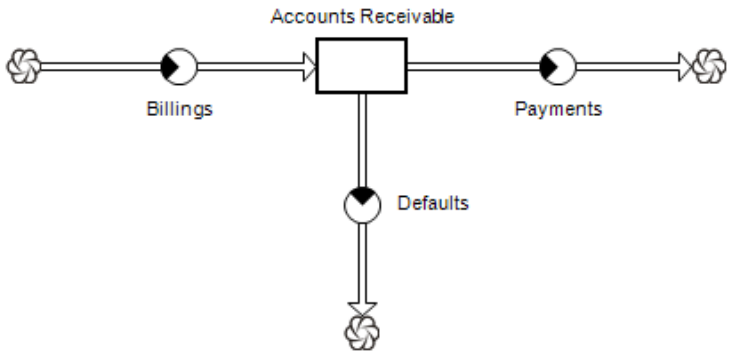


Figure 6.4.: Stock-and-Flow Diagram notation according to Sterman (2000)

In the given example, the stock of receivables is increased by billings and decreased by payments received and by defaults. The flow of billings is conserved in the sense that an invoice remains on the stock until it explicitly flows out via one of two flows. Over time, the rate by which new billings are created or payments are received might change. Therefore, the stock *Accounts Receivable* might also increase or decrease accordingly. However, a decrease of the rate of billings does not necessarily decrease the stock of accounts receivable because a lower (positive) rate still adds more resources to the stock. In addition, the growth of a stock does not only depend on the rate of inflow but also on the ratio of in- and outflows. Because a stock is fully determined by its in- and outflows over time, stock values can be computed by differential equations. Therefore, SFDs can be used for system simulation, thus enabling a double loop learning by providing a virtual world.

To exploit the benefits of both CLDs and SFDs, both types of SD diagrams can be combined. By this way, for example, the variables causing changes to the rates modeled in a SFDs can be modeled by CLDs elements. Together, they form the basis for all kinds of SD models.

6.2. System dynamics in enterprise architecture management

System Dynamics (SD) models have already been used successfully in several disciplines including, for example, project management (Sterman, 2000), economics (Bossel, 2004b) and sociology (Bossel, 2004b). Thus, the goal of this chapter is to provide an overview about SD applications within the Enterprise Architecture Management (EAM) field. Therefore, the results of a literature review as presented by Schneider et al. (2015a) are outlined.

6.2.1. Literature review approach

According to the guidelines for reviewing literature presented by Webster and Watson (2002), the literature review process started by systematically searching relevant databases. In this case, IS journals (AIS senior scholars basket), academic databases (ScienceDirect, IEEE Xplore, ACM DL, GoogleScholar, SpringerLink) and IS conferences (AMCIS, ICIS, ECIS) have been considered. The search was based on the combination of the terms "Enterprise Architecture" or "Enterprise Architecting" and "System Dynamics", "Causal Loop" or "Causal Model" and was limited to title, keywords and abstract. The search revealed 23 initial results. After removing both irrelevant articles and duplicates, seven articles remained. To avoid omitting relevant references, a forward and backward search has been performed subsequently. According to the framework for classifying literature reviews provided by Cooper (1988), this literature review focuses on research outcomes, tries to provide integration, uses a neutral representation and covers only representative articles. Therefore, representative articles for each identified approach are presented in the following section.

6.2.2. Literature review results

To set up an architecture for virtual enterprises, Assimakopoulos and Riggas (2006) present a methodology based on SD. This methodology consists of four steps: identification of business process and interactions, creation of model's structure, testing model using SD software and model evaluation. This methodology is applied to a real world use case in which several SFDs have been developed. According to the authors, "provided a framework for the rapid and efficient integration of the business processes of the participating companies in the virtual enterprise".

Golnam et al. (2010) present an approach to support choice making by integrating SD and their Systemic Enterprise Architecture Methodology. The proposed approach consists of six major steps. In the first step, a conceptualization of the as-is architecture of the enterprise is developed complemented by enterprise models and SFDs. Based on a given scenario, these models are used for simulation. If the simulation yields a problem to be addressed, potential solutions are generated and one of them is selected. Accordingly, a to-be architecture for the enterprise is developed, the SFDs are modified and simulation is performed again. If the potential solution solves the identified problem, the process comes to an end. The approach is demonstrated by a fictitious example which is inspired by a real world case. Thereby, the authors present concrete SFDs modeling the cash in- and outflows of the company and demonstrate the approach's applicability.

Stressing the importance "to explicate and analyze the whys" of an enterprise architecture Sunkle et al. (2013) propose an approach integrating the use of the i* language[1] and SD models, in particular SFDs. Instead of transforming EA models to SD models as proposed by Golnam et al. (2010), Sunkle et al. (2013) propose to begin with EA models, get i* models via meta-model mapping and then construct SD models over i* models with the help of provided guidelines. Based on a case study the authors demonstrate the methods applicability. Thereby, a concrete SFD is presented.

By applying and extending systems of systems engineering, Wojcik and Hoffman (2006) developed a framework for representing complex enterprise characteristics. The framework suggests a hierarchy of models corresponding to different scales of scope and time for the enterprise focusing on enterprises' interactions with the external world. Such dynamics are modeled using a combined highly optimized tolerance and game theory approach allowing for quantitative analyses.

Rashid et al. (2009) present a unified modeling-based method to help with the evaluation of organization design and change decisions. This method consists of three phases: CIMOSA-based enterprise modeling, CLD modeling and simulation modeling. The method has been evaluated during a case study. After the identification of issues of concern via enterprise modeling, concrete CLDs are created for two such issues in which dynamic behavior seems to be the cause.

Based on the distinction of Ossimitz (2000) between system thinking-1 (quantitative oriented using concepts of stocks and flows) and system thinking-2 (qualitative oriented using mental models and delay loops), one can conclude that current literature trying to integrate SD modeling and EA modeling falls mostly in the first category. Approaches of the second category are currently underrepresented. Nevertheless, current literature offers special purpose SFDs, some CLDs and respective simulations (Schneider et al., 2015a). However, a consistent method of SD model development in the context of EAM is still missing.

6.3. A causal loop diagram modeling application landscape diversity

Given the general suitability of SD models to capture dynamic complexity (Sterman, 2000) and successful applications of SD models to complement EA models (Rashid et al., 2009; Golnam et al., 2010; Sunkle et al., 2013) the goal of this chapter is to develop such model to foster understanding of dynamic behavior influencing diversity of application landscapes. As described by Schneider et al. (2015a) dynamic hypotheses are derived from literature and transferred to CLDs. The individual CLDs are then integrated into one consistent CLD describing various different causal dependencies influencing application landscape diversity. To validate the model's utility, a series of expert interviews is performed. Based on the results, additional guidelines could be derived.

[1] http://http://www.cs.toronto.edu/km/istar/

6.3.1. System dynamics modeling design guidelines

Although SD modeling has a long tradition in science (Forrester, 1961) and already proved to be valuable in practice (Sterman, 2000), practitioners within the EAM discipline have not yet adopted this technique. As outlined by Schneider et al. (2015a), one reason therefore might be the absence of a concrete method for developing SD models within companies. Existing methods, for example the methods described by Sterman (2000) and Luna-Reyes and Andersen (2003), are very general in nature, do not account for specifics within companies and assume the presence of experienced modelers and workshop facilitators. Designing a method consisting of concrete activities, their sequence of execution, a role model, a meta-model and appropriate techniques is a difficult task. In order to get one step further in achieving this goal, Schneider et al. (2015a) derived five design guidelines for such method grounded in general problem solving techniques, EAM context as well as EA artifact requirements.

Guideline 1: Distinguish a divergent and a convergent creation phase

Within the domain of complex problem solving, prominent approaches distinguish a divergent phase from a convergent phase (Isaksen and Treffinger, 1985; Jonassen, 1997). Within the divergent phase as many sensible solutions as possible should be identified without any biases. In the subsequent convergent phase solution candidates are systematically analyzed and inappropriate solutions are rejected. Therefore, it becomes vital that input data for SD modeling is collected in a decentralized way. Conducting workshops might be inappropriate because the group might then be subject to process-losses common in groups. These include, for example, production blocking (Diehl and Stroebe, 1987), social loafing (Karau and Williams, 1993), social inhibition (Bond and Titus, 1983) and group polarization (Myers, 2009). Consequently, a method for SD model generation in the context of EAM has to distinguish a divergent from a convergent phase during data collection.

Guideline 2: Gather input from heterogeneous people

There is empirical evidence that the development process for solutions to difficult problems, especially in ecology and psychology, becomes better the more people try to solve it due to their different views implied by their diverse backgrounds (Surowiecki, 2005). Application landscape design is about coordinating the architectural development across levels and departments in an organization (Weiss et al., 2013); hence, a multitude of stakeholders is involved. To include relevant dynamic behaviors accurately, SD models have to be developed based on the opinions of as many diverse viewpoints as possible. While single individuals might not be able to describe a certain behavior of the enterprise objectively and correctly, a group of people is more likely to do so. A good example for the inappropriateness of homogeneous input for SD modeling is described by Sterman (1994). Several people working in functions contributing to order fulfillment have been interviewed about their company's supply chain and all of them had mental models completely overestimating the impact of order fulfillment. If more diverse people had been asked, including customers, accountants and suppliers a more accurate view of the supply chain would have been derived.

Guideline 3: Model for the purpose of learning

Because every model has a clear purpose, it cannot include all aspects present in real world (Stachowiak, 1973). To ensure a feasible scope and allow for timely results (Sterman, 2000), this holds true also for SD models. Consequently, a SD model cannot be complete (Sterman, 2002) and therefore modelers have to identify and maintain a model boundary. Previous research identified that for EA artifacts, accurate granularity and consistency are more important than actuality and completeness (Farwick et al., 2011). Thus, SD models should focus on feedback loops generally unaccounted for in decision makers' mental models. In addition, if used as communication tool with the intention of teaching, i.e. changing inaccurate mental model, SD models have to respect their related cognitive load (Sweller, 1994). Hence, they should limit both intrinsic and extraneous cognitive load by limiting modeled phenomena as well as refrain from adding irrelevant information, repetitions, numerous references or visual distractions.

Guideline 4: Ensure transparency

Correctness has been identified as an important property of any EA artifact and used data sources as one potential reason for erroneous products (Niemi and Pekkola, 2013). As mental models extracted via interviews are the primary data source during SD model development their correctness cannot be proven. Instead, traceability of the origin of model elements, i.e. variables and their causal effects, needs to be provided to foster comprehensibility and trust (Wegmann, 2002). Therefore, the origin of each causal effect has to be documented in detail. This includes at least the technique and date of elicitation as well as the organizational role of the modeler.

Guideline 5: Validate with data

According to Sterman (2000), validating and enhancing qualitative models by simulation is essential to foster learning. Although system thinking techniques (Senge, 1990) and soft systems analysis approaches (Checkland, 1985) are able to enhance understanding about complex situations, they do not allow conducting experiments. Especially if experimenting is infeasible in the real world—as it is for application landscape design—simulations become a suitable tool to understand complex system mechanisms without relying on feedback from real world; thus, enabling faster learning. Although an SD model does not need to be able to reproduce any historic data (Sterman, 2000), data can be used to demonstrate that a model is able to show, for example, general trends.

6.3.2. Modeling approach

Following the advice for SD model development given by Homer and Oliva (2001), the modeling approach starts by deriving dynamic hypotheses. While there exists a broad range of suitable data-gathering techniques (see Luna-Reyes and Andersen, 2003), here the initial set of dynamic hypotheses is derived from literature and personal experience gained from several action research endeavors (Buckl et al., 2013; Sein et al., 2011). As described by Schneider et al. (2015a),

the initial brain-storming session of the author team was performed without interaction. The identified hypotheses were then discussed and harmonized. Thereby, the process adhered to guideline 1 (see Chapter 6.3.1). Each hypothesis was then substantiated by respective literature. Subsequently, for each identified dynamic hypothesis a visual representation by means of a CLD has been created. Because each part of the developed CLDs is therefore linked to its source in literature, required transparency is provided (guideline 4). After the identification of five relevant dynamic hypotheses, the hypothesis generation process has been stopped to allow readers to learn from the resulting model (guideline 3) without being overwhelmed. Because each hypothesis can be mapped to a different role within an organization heterogeneity of perspectives is ensured (guideline 2). However, due to missing data a simulation of the model is not performed (guideline 5) and left for future research.

6.3.3. Dynamic hypotheses

As described in the previous section, five dynamic hypotheses have been derived from literature and personal experience. Each of these hypotheses is outlined in the following and complemented with the respective CLD.

Network effect

If people focus on one single activity, they increase their skill disproportional to those focusing on a set of diverse activities (Page, 2011). This includes, for example, system administrators operating a specific database product instead of operating a set of diverse products. Therefore, if an organization employs highly qualified system administrators, the respective products receive an increasing attractiveness for both solution and enterprise architects due to the availability of efficient operations. Accordingly, although a set of alternative solutions, e.g. database products, is part of the application landscape, a few products will increase in their number of instances over time. This observation is also in line with path-dependence identified by Fürstenau and Kliewer (2015). Figure 6.5 shows the respective CLD for the network effect hypothesis.

The CLD depicted in Figure 6.5 can be read as follows. The lower the diversity within the application landscape, i.e. the higher the degree of standardization, the easier it is for IT operators to increase their skills for a certain technology. An increasing skill level of IT operators leads directly a higher productivity. When solution architects have to make technology selections, technologies for which highly productive IT operators are available are preferred. The more often these products are used, the higher the degree of standardization gets. Consequently, a reinforcing feedback loop exists driving standardization within application landscapes.

Technological progress

Respective literature indicates that a high degree of standardization within the application landscape has a declining effect on the scope of technical possibilities (Sharpe, 1983). As technology diffusion proceeds slowly and therefore its effects are likely to materialize after considerable delay (Baron and Schmidt, 2013), the notifiable impact of standardization on technical possibilities

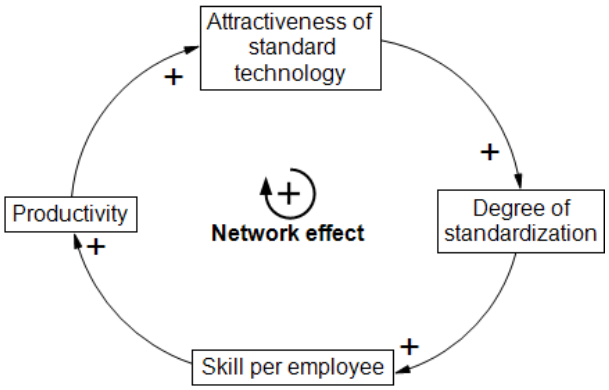


Figure 6.5.: CLD modeling network effect

is also delayed. Once the narrowing technological scope has been recognized, the attractiveness of non-standard technologies increases (Richen and Steinhorst, 2005). Thus, the diversification pressure increases accordingly yielding to less diversity within the application landscape. Together, these causal relationships create a balancing feedback loop as visualized in Figure 6.6.

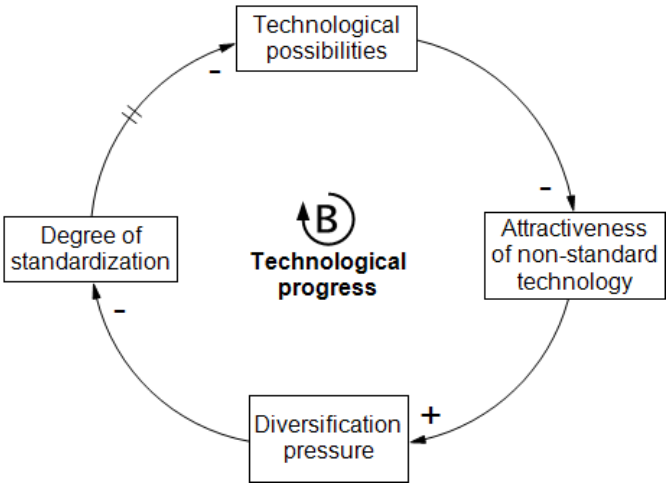


Figure 6.6.: CLD modeling technological progress

The CLD depicted in Figure 6.6 can be read as follows. If the degree of standardization within the application landscape increases, after a considerable amount of time the technological possibilities decrease due to continuous development of new IT products. As a result, solution architects begin to favor non-standard technologies to exploit capabilities of new products. This in turn yields an increase in pressure for diversification which decreases the degree of standardization.

Shadow IT

Aside from the direct effect on diversity, diversification pressure also impacts the incentive of business departments to develop local solutions, i.e. shadow IT (Rentrop and Zimmermann, 2012), which are not corresponding to defined standards. Reasons therefore include, for example, increasingly tech-savvy users, easy access to web-based solutions and available end user computing tools (Barker and Fiedler, 2011), but also misalignment of business demands and provided IT solutions (György et al., 2012). Hence, with a rising number of local, i.e. not centrally managed, solutions the actual standardization degree within the application landscape declines although enterprise architects might not notice the decline. A CLD visualizing this dynamic hypothesis is shown in Figure 6.7.

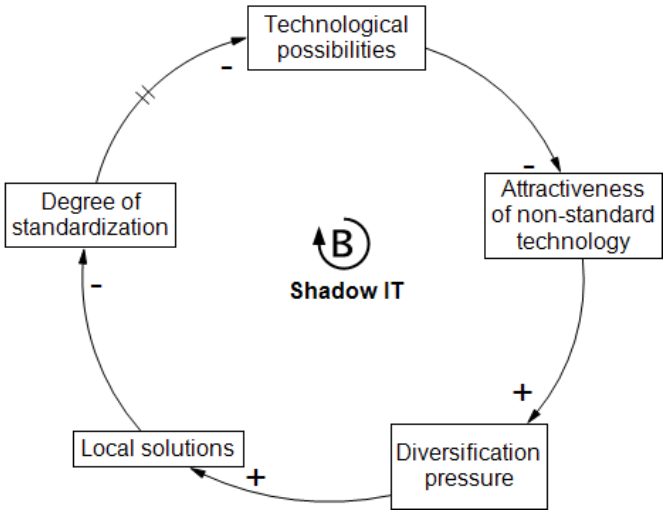


Figure 6.7.: CLD modeling shadow IT

The CLD depicted in Figure 6.7 can be read as follows. If the degree of standardization within the application landscape increases, after a considerable amount of time the technological possibilities decrease due to continuous development of new IT products. As a result, solution architects begin to favor non-standard technologies to exploit capabilities of new products. This in turn yields an increase in pressure for diversification. Unlike in Figure 6.6, the increased pressure for diversification causes an increase in local IT solutions. Accordingly, the degree of standardization is reduced implicitly.

IT cost cutting

Decreasing application landscape diversity and therefore increasing the degree of standardization has been identified as one means to decrease IT costs (Richardson et al., 1990; Richen and Steinhorst, 2005; Boh and Yellin, 2007). Similar to the previously described network effect, the more IT cost savings could be realized, the more requests for standardization are brought forward. Therefore, IT projects and solution architects are constrained with regard to available

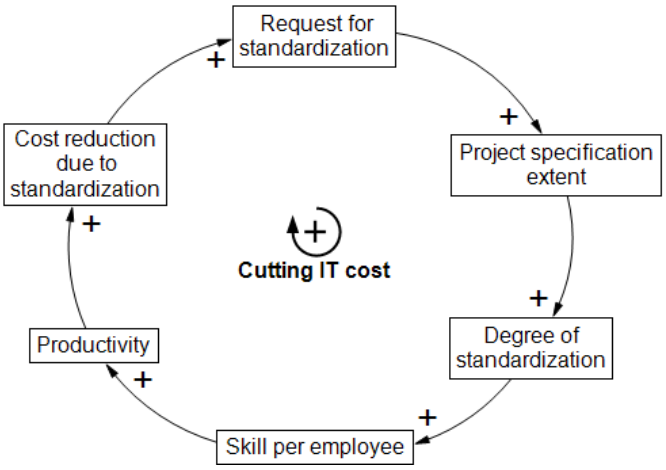


Figure 6.8.: CLD modeling IT cost cutting

technologies which in turn increases the degree of standardization over time. These causal relationships form a reinforcing feedback loop which is visualized in Figure 6.8.

The CLD depicted in Figure 6.8 can be read as follows. As described before, an increase in the degree of standardization leads to an increase in employees' skill levels which in turn increases productivity. The increasing productivity enables the realization of IT cost savings. This success then drives the number of requests for additional cost savings because the approach proved to be successful. A typical way to achieve standardization is an increase in the extent to which technology choices in projects are constrained. Finally, the degree of standardization is thereby increased.

Maintaining the decision-scope

Due to high switching costs and network effects, lock-in effects hinder enterprises from changing suppliers in response to both predictable and unpredictable changes in efficiency (Fürstenau and Kliewer, 2015; Farrell and Klemperer, 2007). Reasons therefore can be, for example, increased specialization of employees and incompatible technologies. The augmenting dependency on certain technologies thus limits the respective decision scope of solution architects. As this limitation is only recognized after a considerable amount of time, the actual perception of the limited decision scope occurs after a certain delay. Subsequently the pressure on diversification is increased (Richen and Steinhorst, 2005) yielding an increase in the degree of standardization. A CLD showing this balancing feedback loop is depicted in Figure 6.9.

The CLD shown in Figure 6.9 can be read as follows. An increase in the degree of standardization causes an increase in employees' skill level. This in turn increases the dependence on certain technologies due to lock-in effects. Thereby, the decision scope regarding the use of new technologies gets limited. After a certain time delay, the perceived limitation of the decision scope increases. Likewise, the pressure for diversification increases which yields a decrease of the degree of standardization within the application landscape.

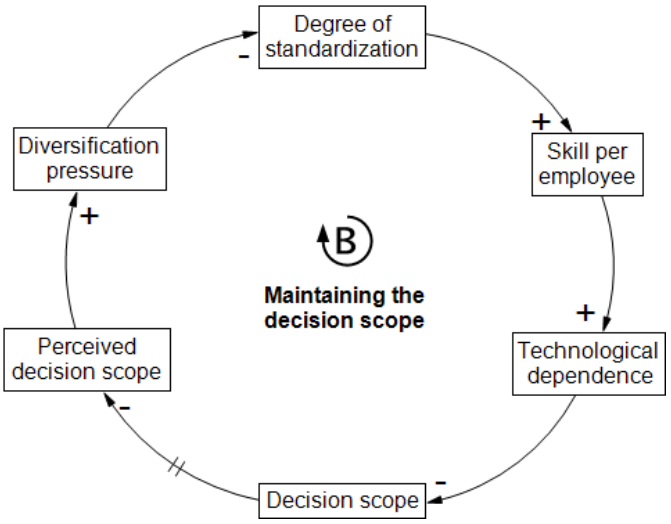


Figure 6.9.: CLD modeling maintenance of decision scope

6.3.4. Integrated CLD model

The next step towards developing a CLD modeling dynamic hypotheses about application landscape behavior with respect to its diversity is to integrate the previously presented CLDs. Therefore, recurring variables need to be unified and causal relationships need to be integrated. Figure 6.10 shows how the different phenomena described in the previous sections interrelate and simultaneously affect the degree of technological standardization within application landscapes. Therein, various colors are used to demarcate different dynamic hypotheses.

Although quite simple in its notation, the CLD depicted in Figure 6.10 is not as understandable as the CLDs modeling only one phenomenon. Nevertheless, it shows how the different phenomena interrelate with each other. In addition, one can see easily that numerous feedback loops of different types (reinforcing and balancing) influence the diversity of application landscapes. In addition, different perspectives are included, for example, perceptions of solution architects, IT operators and external influences like the technological progress. To which degree the individual dynamic hypotheses described in previous sections and the integrated CLD shown in Figure 6.10 conform to actual perceptions of practitioners is evaluated in the next chapter.

6.4. Model evaluation and additional guidelines

As outlined by Schneider et al. (2015a), the goal of this evaluation is to assess whether the dynamic hypotheses modeled in the previously introduced CLD (see Figure 6.10) conform with actual observations by experienced practitioners and whether it can serve as a tool to foster communication among different stakeholders of application landscape design.

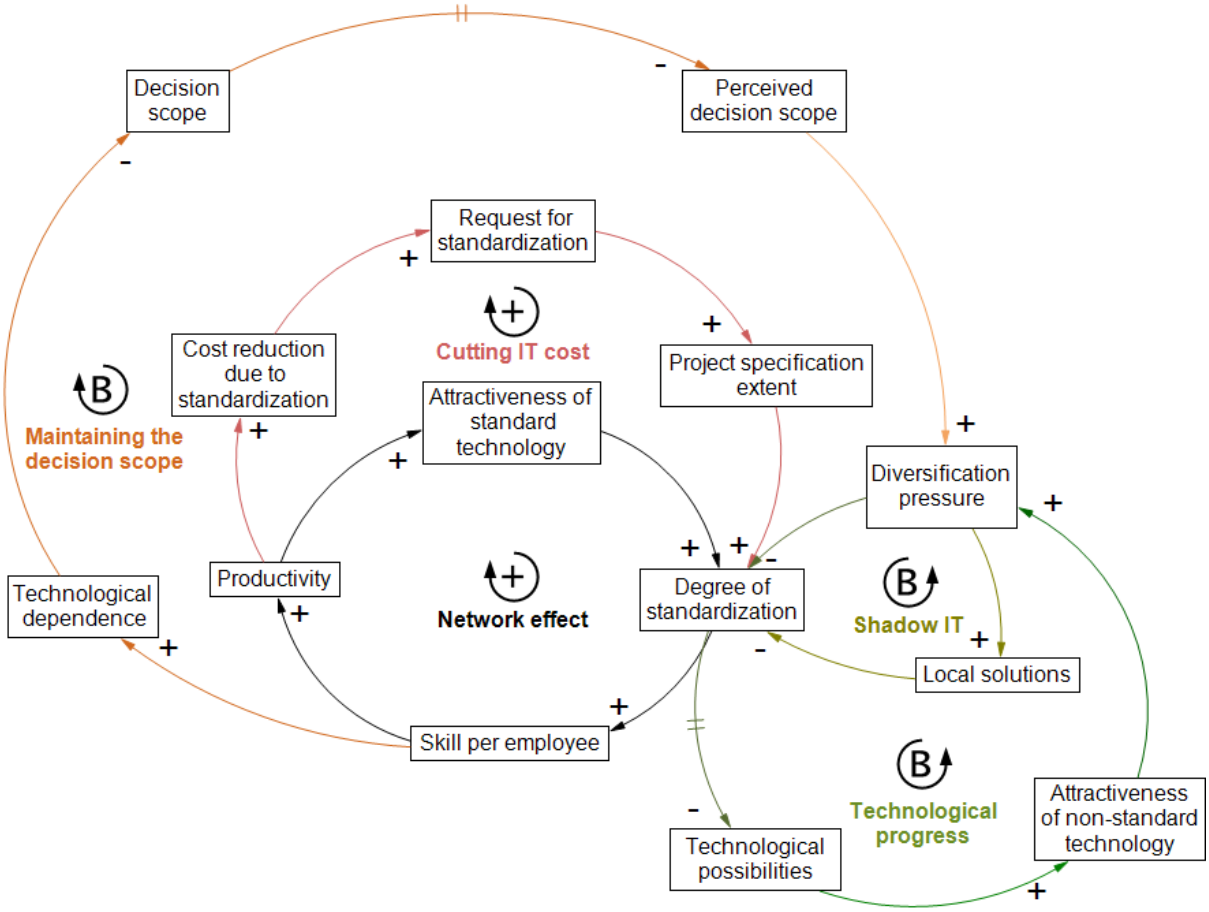


Figure 6.10.: Integrated CLD for application landscape diversity

6.4.1. Evaluation setting

The evaluation of the CLD presented in Figure 6.10 is performed by means of expert interviews (see Bogner et al., 2009) which is a suitable method to validate and improve mental models contained in CLDs (Ford and Sterman, 1998). Therefore, according to the evaluation framework proposed by Cleven et al. (2009), this evaluation approach is qualitative, evaluates a model, follows the epistemology of interpretivism and considers the artifact against the real world.

In total, eight interviews with experienced practitioners have been conducted. Table 6.1 provides an overview about the interviewees' backgrounds. None of the interviewees was familiar with SD in general or CLDs in particular. However, all interviewees were familiar with standardization efforts currently taking place in their organization.

As depicted in Table 6.1, experts from different industries and different levels of experience have been interviewed. Although the sample is not representative, statements of high quality can be expected as respondents have diverse backgrounds (Surowiecki, 2005) and are faced

Id	Role	Industry	Experience
1	Enterprise architecture consultant	Insurance	3 years
2	Project manager (business)	Automotive	> 10 years
3	Project manager (IT)	Gas industry	6 years
4	Project manager (business)	Automotive	3 years
5	Head of Sales & Marketing Analytics	Pharma	> 10 years
6	Enterprise architect	Automotive	2 years
7	IT revision	Service industry	1 year
8	Solution architect	Automotive	> 10 years

Table 6.1.: Overview about interviewed experts from industry (Schneider et al., 2015a)

with standardization issues during their daily work. Interviews have been conducted in a semi-structured way following a questionnaire divided into three major sections: an assessment of the interviewee's general point of view on application landscape standardization, a stepwise presentation and evaluation of the individual dynamic hypotheses and an evaluation of the modeling notation. Interviews have been conducted face-to-face or via video conference tools and lasted between 57 minutes and 93 minutes. During interviews the interviewees have been asked to focus on their own experience to allow for identifying individual examples for explanations. Due to the broad topic under investigation, the advice given by Goguen and Linde (1993) was followed and the last part of the interviews was done in conversation-style to get a deeper understanding of difficult aspects.

6.4.2. Interview results

At the beginning of the structured part of the interviews, a short introduction not longer than five minutes—similar to Chapter 6.1.2—about the notation of CLDs has been given to respective interviewees. Subsequently, based on concrete examples questions about CLDs have been asked to assess whether the interviewee is able to read and interpret a CLD. Surprisingly, all eight interviewees demonstrated their understanding of the used notation. They were able to answer questions about dynamic hypotheses, positive and negative impacts as well as transitive impacts correctly. Furthermore, all of them were able to transfer the presented hypotheses to their individual experiences. However, in all cases a verbal explanation of the respective model was necessary to guide the interviewee through the presented model. This observation is in line with Sterman (2000) who posited that a SD model should never be used without a textual description. In some cases interviewees did not consider time delay symbols correctly.

When thinking about presented dynamic hypotheses, most interviewees began to speculate about the organizational role mostly concerned about the presented phenomena. Likewise, every expert used his personal experience and role understanding for interpreting the presented models. Therefore, it can be concluded that role information is important for these interviewees.

Regarding the confirmation of presented dynamic hypothesis surprisingly all interviewees agreed with every presented CLD although the strength of individual phenomena was ranked differently.

Statements like "*yes this behavior can take place in an organization but in my organization respective countermeasures are in place*" were therefore quite common during interviews. The influence factors the interviewees considered to have more impact obviously correlated with their organizational role and therefore their mental model of the system. This observation is in line with observations made by Sterman (1994).

In two cases, a significant change of the interviewee's mental model could be observed. Before the dynamic hypotheses have been presented each interviewee was asked to describe the causes and effects of application landscape standardization. In these two cases the interviewee mentioned only one or two. When asked the same question after presenting the CLDs, both stated that they would consider the presented additional effects as well. In both cases, the work experience of the interviewees was less than five years.

After the introduction and discussion of each dynamic hypotheses, the interviewees have been confronted with the integrated CLD depicted in Figure 6.10. At first glance, most of the interviewees were overwhelmed by the model. However, after associating the already presented model parts with the integrated model, all of them understood the integrated model as well. When asked for benefits of the integrated model, "getting a general overview" and "getting a consolidated view on a single variable displaying all identified impacts and effects" have been mentioned frequently. Nevertheless, the missing "single message of the diagram" was also mentioned twice.

When asked for potential applications of the presented CLD, all interviewees mentioned the largest benefits for communicating within diverse stakeholders. Thereby, single loop diagrams as well as the integrated model would be useful if combined. Nevertheless, three experts limited potential target groups by excluding top managers due to the diagrams' specific notation.

In conclusion, the content of the integrated CLD, i.e., the presented dynamic hypotheses, was confirmed by the experience of different interviewees. Furthermore, the different CLDs are expected to be able to foster communication in addition to commonly used EA artifacts to create shared understanding among application landscape designers.

6.4.3. EAM-specific modeling guidelines

In addition to the modeling guidelines already described in Chapter 6.3.1, three new guidelines could be derived from previously described interview results and additional comments made during interview sessions. These guidelines are described in the following.

Guideline 6: Explicitly include roles

As mentioned by different interviewees, in presented feedback loops usually different people are involved. Therefore, the model would be easier to understand if the respective organizational roles are included in the diagram. On the one hand, the reader could then faster locate feedback loops relevant for him or her which provides a good starting point to understand the whole diagram. On the other hand, if feedback loops are doubted, additional information about con-

cerned roles could remove such doubts. Therefore, an additional layer should be used for CLDs indicating concerned organizational roles via color-coding or additional icons.

Guideline 7: Use consistent terminology and provide definitions

Four out of eight interviewees pointed out that it is essential to use a consistent terminology in CLDs especially if they are used as means for communication among different language communities. This not only implies usage of organization-specific vocabulary but also the provision of detailed definitions of variables and ideally also causal relationships. To achieve this, a unified glossary can be used.

Guideline 8: Limit the number of modeling elements for presentation

Although modeling elements like the delay symbol are required to indicate system behavior correctly and foster formal modeling, some of them should be excluded from model visualizations if used for communication. This includes, for example, the delay symbol as well as symbols indicating loop direction and type. During the performed interview sessions it became apparent that some people struggle with these elements when trying to understand a CLD. Two interviewees mentioned that they feel distracted by these elements because there is no important information in them. Therefore, a configurable visualization should be used instead of presenting a complete or a formal model.

CHAPTER 7

Artifact evaluation

"What I mean by the word quality *cannot be broken down into subjects and predicates. This is not because* quality *is mysterious but because* quality *is so simple, immediate and direct".*

Robert Piercing, *Zen and the art of motorcycle maintenance, 1974*

7.1. Evaluation approach

In this chapter, the management support system consisting of a conceptual framework of application landscape diversity (see Chapter 4), corresponding indicators (see Chapter 5.4) and respective software implementations (see Chapter 5.5) is evaluated. In line with Franz and Robey (1984) who suggest the use of ideographic rather than nomothetic research strategies for the information systems field and inline with Benbasat et al. (1987) who identified case studies as one particular suitable ideographic method, the results of four different case studies are reported. Conducting a case study to evaluate the developed management support system is appropriate because this research is still at an early, formative stage (Roethlisberger, 1977) targeting a practice-based problem where the experiences of the actors are important and the context of action is critical (Benbasat et al., 1987), e.g., during the development of appropriate distance functions (see Chapter 5.4.4). Accordingly, the artifacts described in this thesis are evaluated with respect to the utility provided in solving actual problems of practitioners (Hevner et al., 2004). Based on the framework for evaluating design science research artifacts proposed

by Cleven et al. (2009), the evaluation approach described in this chapter can be classified as follows. The methods and constructs are evaluated qualitatively following the interpretivism because it is assumed that the evaluation results depend on the individual characteristics of the evaluating subject. Furthermore, the evaluation pursues a knowledge generation function and uses case studies and prototypes as methods. The evaluated objects are the artifacts themselves and the position of the researcher performing the evaluation is internally, i.e., it is carried out by the designer of the artifacts after its implementation. Figure 7.1 summarizes the classification of the evaluation approach described in this chapter.

Variable	Value				
Approach	Qualitative		Quantitative		
Artifact Focus	Technical		Organizational		Strategic
Artifact Type	Construct	Model	Method	Instantiation	Theory
Epistemology	Positivism		Interpretivism		
Function	Knowledge	Control	Development	Legitimization	
Method	Action research	Case study	Field experiment	Formal proofs	
	Controlled experiment	Prototype	Survey		
Object	Artifact		Artifact construction		
Ontology	Realism		Nominalism		
Perspective	Economic	Deployment	Engineering	Epistemological	
Position	Externally		Internally		
Reference Point	Artifact against research gap	Artifact against real world	Research gap against real world		
Time	Ex ante		Ex post		

Figure 7.1.: Instantiation of the evaluation framework proposed by (Cleven et al., 2009)

In the following chapters, four different case studies are described. First, the disparity metric is used to support decisions about technical platforms for a single business application at a large financial service provider. Second, the disparity and adolescence metrics together with the prototypical implementation and the system dynamics model are used to justify increasing IT costs at a mid-sized manufacturing company. Third, the adolescence metric together with the prototypical implementation are used to support enterprise architects at a large insurance company to assess diversity within their application landscape. Fourth, the disparity metric is used to support a diversification decision regarding technology components at a large car manufacturing company. Table 7.1 provides an overview of these case studies and the artifacts evaluated in each case. Although the findings might not be fully generalizable, the participating companies exhibit some variety with regard to their size as well as their industry sector. However, all operate on a global scale.

Case study	Disparity	Adolescence	Implementation	System dynamics	Industry sector	FTEs
Case 1	✓				Financial services	>100,000
Case 2	✓	✓	✓	✓	Manufacturing	>20,000
Case 3		✓	✓		Insurance	>21,000
Case 4	✓				Automotive	> 230,000

Table 7.1.: Overview about case studies and evaluated artifacts

7.2. Case 1: Supporting single application design at a large financial service provider

In this case study, the company under investigation was facing a difficult decision regarding the identification of a suitable technical platform for one of their business applications. To support this decision, the complexity and therefore also the diversity of two potential platforms should be taken into account. Accordingly, the goal of this case study is to evaluate in how far the disparity metric presented in Chapter 5.4.4 can provide useful information in such case.

Organization: The respective company is a globally operating financial service provider headquartered in Italy. While IT operations have been carved-out to a service company, the responsible enterprise architects are part of the holdings central IT department. In 2013, this company employed more than 100,000 people and achieved a revenue of more than 10 billion Euro. Some years ago the company went through a large merger with another financial service provider. Since then, the EAM team is distributed across different countries. The case study described here is conducted with architects responsible for the technical architecture of the company. This function has been established many years ago and all participating architects have more than 10 years of relevant professional experience. To store architectural information, the company uses one of the market leading software applications. Like other financial service providers, this company is facing cost pressure in particular in the IT department. Accordingly, a recently raised proposal to move the EAM tool to a new technical platform was brought into question. Beside functional and non-functional requirements the complexity of the new platform should also be considered due to the goal to reduce the overall complexity of the application landscape. However, the company does not have appropriate means to quantify complexity or diversity of technical platforms.

Course of action: To manage the technical components, i.e. components used to build business applications which are independent of any business requirements such as operating systems or database management systems, the architects developed a comprehensive taxonomy to classify such components. On the lowest level, individual technical elements are modeled. On the next level valid and admissible technical components are defined comprising hardware, software and

facilities. One or more of such components are then integrated to a technical platform. In case of the EAM tool, two different technical platforms exist which could be used as a foundation. First, together with three architects a suitable approach to assessing the complexity of such technical platforms has been developed. As input, the complexity and diversity indicators described in Chapter 5 as well as a comprehensive study performed by the company in 2012 have been used. It has been decided that counting the amount of technical elements used to build a platform is sufficient to assess complexity. Accordingly, only the *variety* aspect of diversity is regarded. However, the architects noticed that counting technical elements is not a suitable means to compare all kinds of platforms. If a platform offers more capabilities, it is likely to require more technical elements. Accordingly, it needs to be ensured that two platforms are relatively equal in terms of their capabilities in order be comparable by counting their technical elements. Therefore, the *disparity* of technical platforms has to be considered additionally. As described in Chapter 5.4.4, the disparity metric is always context-sensitive and therefore cannot be applied to arbitrary technical platforms. Fortunately, the company already uses a classification schema for their technical platforms. As visualized in Figure 7.2, it distinguishes between infrastructure, platform and software services as well as presentation, data, business logic and other layers. One cannot assume that to provide each kind of capability modeled by this matrix an equal amount of technical elements is required. Therefore, if a technical platform provides additional or other capabilities the number of technical components might be biased significantly. However, if two technical platforms provide the same capabilities, the number of technical elements as complexity indicator can be compared. To assess how different the provided capabilities of two platforms are their *disparity* needs to be assessed. As presented in Chapter 5.4.4, this requires a distance function for technical platforms. Here, after consultation with the company's architects, it has been decided that the distance function should be based on the matrix shown in Figure 7.2. Each cell of this matrix describes a capability which can either be fulfilled or not. To calculate the degree to which two platforms provide the same capabilities, the Jaccard (1912) coefficient can be used. Based on this distance function the Weitzman (1992) metric can be calculated accordingly. However, this second step is trivial because only two platforms are considered and therefore the metric result is equal to the distance function.

	Presentation	Logic	Data	...
SaaS				
PaaS				
IaaS				

Figure 7.2.: Matrix to describe technical platforms (excerpt)

In the actual case, two different technical platforms (TP1 and TP2) were analyzed. Both platforms included technical elements for data storage, operating systems and data transfer. TP1 was composed of five technical elements covering presentation, logic and data on the IaaS and PaaS layer of the technical platform matrix. In contrast, TP2 was composed of nine technical elements covering also presentation, logic and data on the IaaS and PaaS layer of the technical platform matrix. Accordingly, the complexity of TP1 is defined as five and of TP2 as nine; thus

TP2 is more complex. The Jaccard coefficient defined as $\frac{TP1 \cap TP2}{TP1 \cup TP2}$ for these two platforms is one because they provide equal capabilities according to the technical platform matrix. Due to the obvious absence of disparity, the number of technical components can be used to assess complexity of TP1 and TP2. Accordingly, TP2 is more complex than TP1. An in-depth analysis revealed that the reason for this difference in complexity is mostly driven by the ability of TP2 to store data in a cluster rather than on a single server thus providing high availability.

Results: In a final workshop, in which four IT architects as well as the head of the technical architecture division participated, the results of the complexity and disparity assessment for the concrete case of the technical platform used to run the EAM tool were discussed. All participants considered the information provided by the applied indicators to be very helpful with regard to the upcoming decision. Assessing disparity to ensure that the results of the element counting metric are actually comparable was especially welcomed. Hereby, the easiness and comprehensibility of the calculation was named as additional benefit. Therefore, the IT architects decided to implement this indicator in their currently used EAM tool and calculate it also for future decisions. However, they planned to replace the quite abstract matrix by a more detailed model to increase the expressiveness of the metric in the future. All in all, the company decided to use this approach also for future decisions and to work on an extended approach. Thus, the question addressed by this case study whether the disparity metric presented in Chapter 5.4.4 is able to provide relevant information for application landscape design decisions can be answered positively based on the presented case.

7.3. Case 2: Supporting standardization at a mid-sized manufacturing company

In this case study, the company under investigation wanted to assess the complexity and diversity of the application landscape in order to justify a recent cost increase of the IT on the one hand and identify potential for complexity reduction on the other hand. To provide the required information to senior managers, all artifacts presented in this thesis including the adolescence metric (Chapter 5.4.3), the disparity metric (Chapter 5.4.4), the prototypical implementation (Chapter 5.5) and the system dynamics model (Chapter 6.1.2) were used. Accordingly, the goal of this case study is to evaluate to which degree these artifacts provide utility in the described context.

Organization: The respective company is a globally operating producer of different kinds of optical goods and machines headquartered in Germany. In 2014, the company employed more than 20,000 people and achieved a revenue of more than 4 billion Euro. Although the different sectors can independently decide about required IT support to a certain degree, IT governance is centralized. Thus, the company runs a federated approach to IT management. The enterprise architecture function is thereby located in the central IT department and has been established some years ago. Important duties include, but are not limited to, collecting architectural data, providing respective reports and assessing the application landscape from a company-wide rather than sector-specific view. This case study was conducted mainly with the chief enterprise architect.

Indicator	Source
Number of interfaces per application	Schneider et al. (2015c)
Degree of deviation from standard OS	Mocker (2009)
Degree of deviation from standard DBMS	Mocker (2009)
Heterogeneity of OS	Schuetz et al. (2013)
Heterogeneity of DBMS	Schuetz et al. (2013)
Number of application	Schneider et al. (2015c)
Customization level	Schneider et al. (2015c)
Number of technical component	Schneider et al. (2015c)
Adolescence of OS	Chapter 5.4.3
Adolescence of DBMS	Chapter 5.4.3
Disparity of applications	Chapter 5.4.4

Table 7.2.: Complexity and diversity indicators selected for implementation in case study two

Course of action: Based on the assumption that IT complexity drives IT costs (see Chapter 3.2.3), the first step in this case study was to select appropriate indicators able to quantify complexity and diversity of the company's application landscape. Therefore, the set of indicators presented in Chapter 5 was analyzed with regard to several criteria based on the organizational context of the company.

These criteria include:

- scientific soundness
- known use cases (implementations in practice)
- availability of data within the company
- feasibility of implementation
- comprehensibility
- subjective rating of the enterprise architect

By this approach, the initial set of 21 indicators could be reduced to eleven. For the remaining eleven indicators several implementations are known, data is available via the EAM tool, implementation seems to be feasible, comprehensibility is either easy or moderate and the subjective rating was positive. Table 7.2 provides an overview of the selected indicators.

As can be seen from Table 7.2, the disparity metric as well as the adolescence metric fulfill the criteria of the company and were selected by the enterprise architect due to their promising decision support. Interestingly, the disparity metric was considered to be useful for assessing redundancy of business applications within certain domains rather than for technical components (see Chapter 7.2). Although suitable also for other kinds of technical components, operating systems and database management systems have been chosen as a first step towards adolescence calculations, because for those the required data was available. In the following, only these three metrics are considered because the others are irrelevant for this evaluation.

Operating system	V1	V2	V3	V4	V5	V6	Adolescence
OS1	21	43					-0.34
OS2	1						-1.00
OS3	7						-1.00
OS4	6						-1.00
OS5	1						-1.00
OS6	2						-1.00
OS7	19	2					-0.81
OS8	1	25	27				-0.49
OS9	8	72	959	303	145		-0.30
OS10	3	1					-0.50
OS11	4						-1.00
OS12	1	1	4	5	4	2	-0.18
OS13	1	0					1.00
OS14	1						-1.00
OS15	3	77	1				0.02
OS16	2	3	2	2	2		0.05

Table 7.3.: Distribution of operating system instances among versions

First, the adolescence of operating systems was assessed by calculating the respective indicator as described in Chapter 5.4.3 based on a snap shot of the EAM tool's database. Thereby, the prototypical implementation described in Chapter 5.5 was used which is later described in detail. In total, 16 different operating system product lines were in use. For seven of these, only one version was currently in operation which was considered to be the most recent version usable. Therefore, there is no adolescence of these components and the indicator results per definition in -1. However, for the other nine components different versions were in use. Table 7.3 shows these versions with the respective number of instances. As one can easily see from the respective numbers, adolescence can be observed especially for OS13, OS15 and OS16. The negative sign of the other components indicates that more new versions than old versions are in use. After presenting this result to the company, this observation was confirmed by the enterprise architect. Thereby, the adolescence indicator results made it easier to identify operating systems currently subject to adolescence which relates to the *balance* aspect of diversity.

As one can see easily, a plain visualization of the distribution of versions for each operating system is not sufficient to identify operating systems having high adolescence. Figures 7.3 and 7.4 show exemplarily the distributions of versions belonging to OS9 and OS12. If the enterprise architect wants to initiate the migration of old versions for the operating system having most adolescence, such representation makes it difficult to select the right one because there is no clear representation of adolescence included.

Likewise, the adolescence of Database Management Systems (DBMSs) was assessed. In total, 18 different DBMS product lines were currently used by the company. For ten of these DBMSs, only one version was in use and therefore adolescence needs not to be considered. For the others, adolescence can be calculated because two or more different versions were in use. Table 7.4

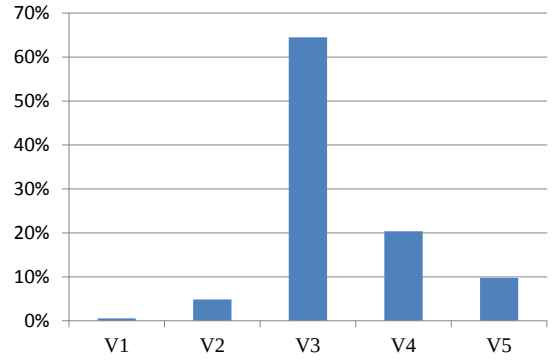


Figure 7.3.: Version distribution of OS9

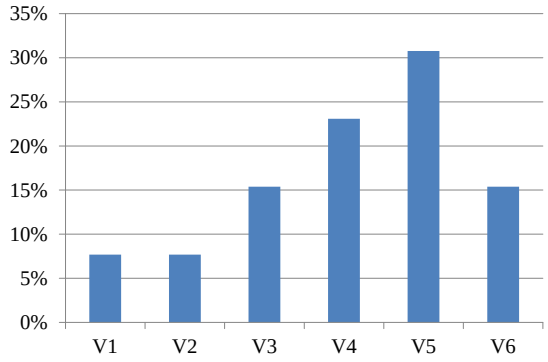


Figure 7.4.: Version distribution of OS12

summarizes the data about DBMSs and their versions in use. By analyzing the adolescence of DBMS components 3, 10, 16 and 17, one can see easily that they are subject to adolescence due to their positive sign. Therefore, for those components instances of old versions need to be migrated to new versions in order to limit their adolescence. In contrast, e.g. DBMS9, DBMS14 and DBMS15 have highly negative adolescence values. Accordingly, most of their versions in use were quite new so there was currently no need for action. Again, the adolescence indicator facilitated the identification of technical components for which more old than new versions are in use by providing a quantitative perspective on their valuated *balance* of their versions.

In the course of the development of a new indicator to quantify potential redundancy among the company's business applications, the disparity metric introduced in Chapter 5.4.4 has been used. Thereby, functional redundancy is assessed based on a matrix which combines the company's domain model with its process model. The arising cells form the capabilities which have to be supported by at least one business application. If more than one application is located in such cell, these applications might be subject to functional redundancy. However, this must not be the case, because some domains might require more than one single business application for a given process. Therefore, the *disparity* of such applications can provide the basis for assessing functional redundancy within a cell of this matrix. If applications are different with respect to a desired characteristic, they are not redundant. If they are considered to be equal, i.e. disparity is 0, they can be considered to be redundant which needs to be proven by an in-depth analysis.

Based on the same snap-shot of the EA repository, the disparity metric has been used in the aforementioned context. In this case, the respective company was using 841 business applications which were grouped into 8 domains and supported 49 high-level business processes. This yields an average of 2.15 applications per capability. Taking into account, that not all domains are relevant for each business process, e.g. supply chain management is a domain not required for order processing, the amount of relevant combinations of business processes and domains decreases to 126 yielding an average number of 6.67 applications per capability. Because for some capabilities, the number of applications is largely above the average, e.g. 31 applications to support supply chain management in logistics processes, the disparity metric was used to assess how different these applications are with respect to desired characteristics. Two main characteristics have been identified in cooperation with the enterprise architect and a demand manager of the company under investigation: country and supplier. The country where logistic

DBMS	V1	V2	V3	V4	V5	V6	Adolescence
DBMS1	4						-1.00
DBMS2	1						-1.00
DBMS3	1	0					1.00
DBMS4	1						-1.00
DBMS5	1						-1.00
DBMS6	1						-1.00
DBMS7	3						-1.00
DBMS8	1	53	103	179	48	5	-0.04
DBMS9	1	27					-0.93
DBMS10	8	56					0.75
DBMS11	8						-1.00
DBMS12	11						-1.00
DBMS13	1						-1.00
DBMS14	2	5					-0.43
DBMS15	2	8					-0.50
DBMS16	2	0					1.00
DBMS17	2	1	1				0.25
DBMS18	2						-1.00

Table 7.4.: Distribution of database management system instances among versions

processes are carried out is a desired characteristic which differentiates business applications from each other in this area because of the different legal requirements which often cannot be fulfilled by a single application. In addition, different suppliers often require different applications for process integration. Taking these two aspects into account, the distance among all business applications could be calculated as described in Chapter 5.4.4. In this case, the total disparity value is not of interest but the single distance values are. Interestingly, only one application had a distance of 0 to another application and can therefore be considered to be functional redundant. All other applications differ in at least one characteristic. Accordingly, this capability is not subject to functional redundancy although one application has to be assessed in detail. This fact could easily be derived by the application of the disparity metric. By the same approach, another capability for which several different applications were used could be assessed. However, in a third case, the calculation of the disparity metric yielded lots of applications having zero distance to other applications. Accordingly, the respective applications are considered to be functional redundant and the company decided to assess the potential for retiring them. After the calculation of several disparity values, the final assessment could be enhanced. The coarse-grained capabilities could be split up into smaller capabilities based on the desired characteristics for disparity. The number of capabilities to be supported rose to 203. Accordingly, the average number of applications per capability decreases to 4.14. Regarding the utility of the disparity metric introduced in Chapter 5.4.4 in this context, both, the enterprise architect as well as the demand manager agreed that "to provide an overview about the potential redundancy within the application landscape, disparity assessments help a lot". Furthermore, the redundancy metric

will be used by the company to steer the application landscape which would be too inaccurate without disparity considerations.

To calculate the previously described indicator results, the prototypical implementation presented in Chapter 5.5 was used. Therefore, the whole application landscape data was exported as CSV file from the company's EAM tool. In this case, the data selection was performed manually by extending the R scripts respectively. On average, this required about 30 min. per indicator and resulted in a reusable solution. The calculation was performed based on the implementations already described. The results were then written to a CSV file again. The resulting file can be used for further processing, if required, or stored to allow historical analyses in the future. Although it would have been desired to see calculation results directly in the EAM tool, this was not possible due to technical limitations of the tool used by the company. Therefore, the stakeholders at the company were satisfied with this implementation approach. However, after the case study, the company decided to implement these indicators with a different technology supported by the EAM tool vendor but according to the same architecture. Thereby, the R language has been replaced by Extensible Stylesheet Language Transformations (XSLTs). The reason for using another language was that the company needed external technical support which is currently not available for the prototypical implementation described in Chapter 5.5. Both approaches have been valued superior to a solution based on Microsoft Excel because they require less manual interaction.

Because increasing standardization and reuse of both technical components and business applications were on the list of goals to be achieved by the EAM function, the System Dynamics model presented in Chapter 6.1.2 was used to reflect on the problem and appropriate strategies. During a half-day workshop together with the enterprise architect responsible for guiding standardization activities, the modeling notation as well as its content were evaluated. First, the existence of the modeled cause-and-effect relationships has been confirmed. For example, due to the technological progress new types of applications and technical components have to be used or are required by the business which increases the application landscape's diversity. This can be seen, for example, in the adolescence of some components described before. However, the shadow IT causal loop is not that strong in the respective company due to the federated model of IT governance. In the past, the network effect combined with the IT cost cutting loop were very dominant in the company. By outsourcing the IT function more and more to an IT service provider, the whole IT landscape was limited to the technologies and capabilities of the providing company. As described by the SD model, this in turn lead to decreasing freedom regarding the decision-scope. Accordingly, the company insourced some crucial parts of the application landscape and searched for a second IT provider to complement the first. In addition to the fact that the CLD model described in Chapter 6.1.2 is able to describe the history of the company's application landscape for the past years with respect to its degree of diversity, the model itself was recognized to be useful also for communication among different stakeholders today. The evaluating enterprise architect speculated that maybe senior managers might be confused by the notation and large amount of content, but some colleagues of other departments might be a suitable audience. In such cases the CLD is expected to provide a holistic understanding of the problem. Finally, the enterprise architect draw the conclusion that neither total standardization nor diversification are desired or achievable and that the degree of diversity will vary over

time. Therefore, it can be concluded that the developed CLD is able to transfer its message and provides utility at least in this case.

Results: As demonstrated above, the adolescence as well as the disparity metric were able to provide useful information to the stakeholders of the company. In a final workshop, the head of IT governance, the enterprise architect and a demand manager all agreed that these metrics are valuable for their work. Finally, the company decided to implement these indicators and perform calculations every three month in order to be able to see trends over time. The implementation approach was similar to one proposed in Chapter 5.5. The set of selected indicators has also been presented to senior IT managers and is likely to be used for application landscape design decisions in the future. Thus, it can be concluded that both indicators, the suggested implementation approach as well as the SD model provided utility in this case.

7.4. Case 3: Supporting technology governance at a large insurance company

In this case study—which is also partly described by Schneider et al. (2016)—the company under investigation was facing the challenge to identify technology components used in their application landscape which are subject to adolescence. These technology components are considered to increase security threats and risks due to accumulated migration efforts. Accordingly, the goal of this case study is to evaluate if the adolescence indicator presented in Chapter 5.4.3 is able to support respective enterprise architects to identify adolescence in their application landscape. Thereby, the prototypical implementation presented in Chapter 5.5 is used.

Organization: This case study took place in a globally operating insurance company headquartered in Germany. In 2013, the company employed more than 21,000 people and achieved a revenue of more than 28 billion Euro. The company consists of a holding which owns several quite independent subsidiaries in different countries. To consistently steer the architecture of the whole group, a significant part of the EAM function is situated in the holding. This case study was performed in cooperation with enterprise architects responsible for the group-wide technical architecture in particular their standards. Therefore, the goal of this department is that as many technical components as possible adhere to the defined set of standards. Furthermore, independently of these standards the department needs to assure that the whole landscape is up-to-date in order to minimize security threats and accumulating efforts for migration. However, the identification of such old components was a tough challenge for the enterprise architects because of several reasons. First, the variety of technical components removes the possibility that a single person or small team of enterprise architects is able to judge if a specific technological component is too old to be used due to missing component-specific expertise. Second, assessing each component individually would be associated with high efforts due to the large amount of different components in use which was considered to be unaffordable.

Course of action: To evaluate if the adolescence indicator presented in Chapter 5.4.3 is able to support the enterprise architects of the respective company in identifying adolescence among the technical components of the application landscape, it was applied to the areas of operating systems and database management systems. Thereby, the enterprise architects were happy to

see that only information about the components, their versions, their predecessor/successor relationships and their number of instances was required because this data was already available and no additional data collection effort was required. To calculate the adolescence indicator, the required data available in CSV format was directly fed into the prototypical software solution presented in Chapter 5.5. Table 7.5 summarizes the provided data about operating systems. Without influencing the results in general, information about some operating systems is not disclosed here due to data protection reasons.

OS	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	Adolescence
OS1	20	235										-0.8431
OS3	243	3	777	2	397	5	6	5	1167	5	32	0.0198
OS4	6	14										-0.4000
OS6	2	1587	1496	1								0.0102

Table 7.5.: Distribution of operating system instances among versions

As can be seen from the adolescence calculation results, there was no adolescence of OS1 because only a few instances of old versions were running. The same was true for OS4. However, OS3 and OS6 exhibit adolescence indicated by the positive sign of the indicator results. Although this impression might be easily derived for OS6 due to its clear distribution slightly skewed to the right, without the indicator it might be difficult to assess adolescence of component OS3.

The same procedure was used to assess DBMS components. Likewise, Table 7.6 summarizes the provided data. However, in this case not always the newest release of a certain component was in use. Therefore, the newest versions were added with zero instances in consultation with the respective enterprise architects if these versions would get the approval for usage. Again, without influencing the results in general, information about some database management systems is not disclosed here due to data protection reasons.

DBMS	V1	V2	V3	V4	V5	V6	V7	V8	Adolescence
DBMS1	8	2	37	100	4	9			-0.0925
DBMS2	16	135	13	16	0	0			0.5356
DBMS3	12	23	8	0					0.3953
DBMS4	8	154	56	331	1				-0.1482
DBMS5	1	135	2	6	30	1	51	2	0.2456

Table 7.6.: Distribution of database management system instances among versions

According to Table 7.6, DBMS components 1 and 4 do not show any adolescence. However, the other DBMS components do show significant adolescence. Especially the DBMS2 instances are mostly old and the two most recent versions are not in operation at all. But also DBMS3 and DBMS5 do show high adolescence values which can be seen by the positive sign and the relatively high absolute value of the adolescence indicator result. To provide an integrated overview about

the DBMS products in use, the visualization presented in Chapter 5.4.3 can be used. Figure 7.5 integrates and visualizes the results of Table 7.6.

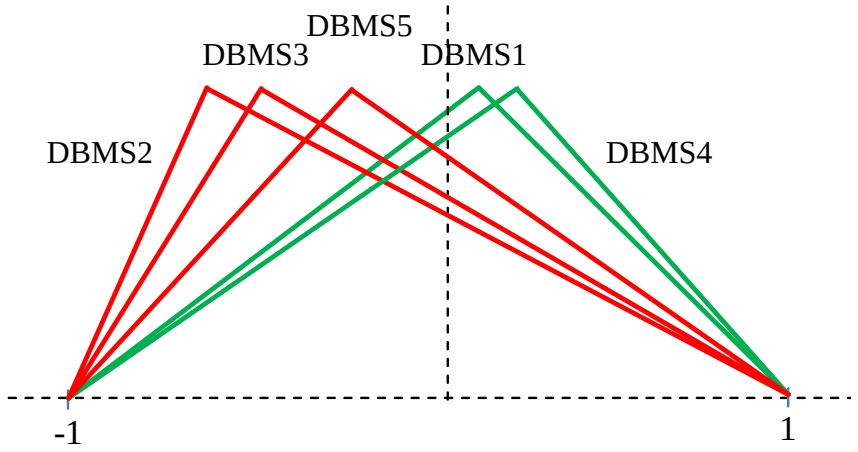


Figure 7.5.: Integrated visualization of DBMS adolescence values

Results: The results of the adolescence analysis have been presented to respective IT architects and IT executives from the central IT department. Thereby, the three questioned experts highlighted that

> "*the normed quantification of adolescence leads to a better comparability of technical domains across subsidiaries in different countries*".

In addition, all experts agreed on that the provided information is useful for them and a future usage of the indicator is appreciated. Furthermore, the quantification facilitates the development of a prioritization of required activities for respective IT components or respective subsidiaries. In addition, the loosely coupled software solution again proved its utility because the calculation of the aforementioned indicators could be done easily based on CSV files as input.

7.5. Case 4: Supporting technology diversification at a large automotive company

In this case study, the participating company was facing a challenge in selecting appropriate technology components to increase the application landscape's diversity in certain areas including application servers and DBMSs. Accordingly, the goal of this case study is to evaluate if the disparity metric described in Chapter 5.4.4 can support this decision.

Organization: The participating company in this case study is a globally operating car manufacturer headquartered in Germany. In 2014, the company employed more than 230,000 people and achieved a revenue of more than 120 billion Euro. The company is divided into several divisions including cars, buses and trucks. Therefore, the IT governance in general and the EAM function in particular follow a federated approach. However, the responsibility for the technical architecture is located at the central part of the IT department. Due to several rea-

Application server	Source access	JEE	Footprint
AS1	Open source	No	Small
AS2	Open source	No	Small
AS3	Open source	Yes	Medium
AS4	Open source	Yes	Medium
AS5	Open source	Yes	Medium
AS6	Open source	Yes	Large
AS7	Closed source	Yes	Large

Table 7.7.: Application server candidate assessment for disparity

sons, the application landscape of the company was highly standardized regarding application servers. Because of emerging demands from business, the one-size-fits-all application server in use was no longer able to fulfill all requirements. Therefore, the EAM team decided to increase the diversity of the application landscape in this area to provide a suitable basis for IT projects. To identify suitable complementary products, the enterprise architects performed an evaluation of products available on the market. Thereby, the disparity should be increased with respect to some desired characteristics. However, the variety should not exceed a certain limit. In this case study, the already performed analysis is repeated by using the disparity assessment approach presented in this thesis to allow a comparison of both approaches.

Course of action: The company under investigation provided a list of nine characteristics which have been used to identify suitable application server products as well as an assessment of seven candidate products including the product currently used. An analysis of these nine characteristics revealed that only three of them are relevant for disparity because several manifestations of these characteristics are desired. This includes, for example, the access to the source code as well as the Java Enterprise Edition (JEE) support. The company would like to have both, closed source and open source application servers to fulfill different demands. Likewise, JEE application servers are still required although not in every case. In contrast, the other characteristics form mandatory requirements which are therefore not related to disparity. For example, a selected product needs to be compliant to several operating systems and development and operations support needs to be available. It is obvious, that for those characteristics only one manifestation is desired and not both ends of the respective dimension, i.e. a product being not compliant to several operating systems currently used by the company. To assess the disparity of the candidate products, each has been assessed based to the characteristics relevant for disparity in this context (see Table 7.7). The footprint summarizes the size of an application server in terms of installation and operating efforts.

Based on these manifestations, the distance matrix for all candidate products has been computed based on Euclidean distances. As visualized in Figure 7.6, a dendrogram has then been used to visualize the distances of all candidate products. The strategy of the respective company was to increase the variety of application servers only slightly, i.e. not more than three different products should be used. Accordingly, the currently used component AS7 should be complemented by a maximum of two additional components. The selection can be facilitated by using the dendrogram. If AS7 is already set and only two components should be chosen which maximize

the disparity among application servers, the dendrogram defines two clusters: AS1 and AS2 form the first cluster and AS3, AS4, AS5 and AS6 form the second cluster. If one component of each cluster is chosen, the disparity is maximized. Thereby, it makes no difference, which component is selected. In addition, the selection process within each cluster can be facilitated by the characteristics initially omitted. Based on the respective assessments, for example, AS1 is preferred to AS2 and is therefore selected within the first cluster. Likewise, AS5 is selected out of the second cluster due to its superiority with regard to the initially omitted characteristics. The set consisting of AS1, AS5 and AS7 forms the recommended selection for application servers in this context based on a maximization of disparity and subsequent consideration of additional quality criteria.

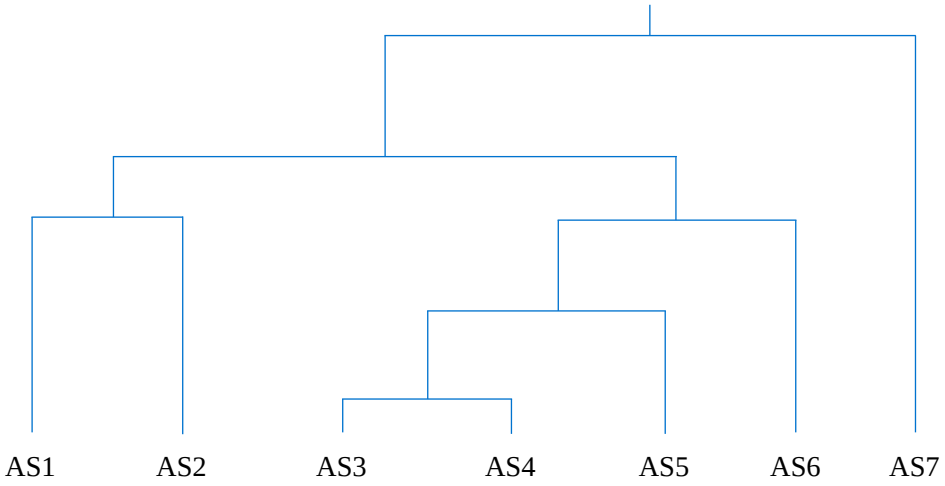


Figure 7.6.: Dendrogram for application server candidate products

Results: The recommendation generated by using the disparity metric equals the result of the unstructured approach carried out by the company under investigation. To assess the utility of the disparity approach, an enterprise architecture consultant who was involved in the initial assessment of the company was interviewed. Thereby, he confirmed that the disparity approach is more comprehensible as well as repeatable. In addition, it seems to be feasible and easy to understand. He highlighted and praised the importance of distinguishing disparity-related aspects form general quality aspects in the context of such decisions. That the results equal those of the initial unstructured assessment adds additional credibility to the approach. However, additional decision support regarding the identification of the appropriate number of clusters was mentioned as a potential for improvement as well as an integrated view unifying the dendrogram and the ranking of their components based on the quality criteria assessment. It can be concluded that the disparity metric proposed in Chapter 5.4.4 performed as well as the company's current approach but provides a more comprehensible and reusable process and visualization to guide decisions increasing an application landscape's diversity.

CHAPTER 8

Conclusion

"If I have seen further, it is because I am standing on the shoulders of giants."

Sir Isaac Newton, *A letter in 1676*

The success of application landscape design is strongly influenced by decisions affecting the application landscape's diversity. Thereby, three different aspects of diversity need to be considered at the concept level: *variety*, *balance* and *disparity*. Due to the huge extent of many application landscapes, enterprise architects and other application landscape designers rely on support systems as, for example, respective indicators to make better informed design decisions. As has been proven by a thorough literature review, currently proposed complexity or diversity indicators fail in addressing all relevant diversity aspects. Practitioners trying to apply and adopt existing indicators experience several difficulties in doing so. This thesis solves the aforementioned problem by presenting the missing diversity indicators as well as a new viewpoint on the problem of standardization following the complex systems paradigm. Recalling the research objective as stated in the introduction, the results of this thesis are presented, a critical reflection on the findings is provided to describe the limitations of this research and suggestions for future research are provided.

8.1. Summary of results

In Chapter 1, the experienced research gap motivated by both, a literature review as well as industry expert interviews, has been discussed and the main research objective of this thesis has been defined as follows:

What kind of decision support do enterprise architects require to design an application landscape with regard to its diversity?

This research objective has been subdivided into five research questions (see Chapter 1.1). These research questions guide the design and structure of this thesis. In the following, the individual contributions of the single chapters are summarized and discussed in order to answer the research questions.

In Chapter 2, the foundation for this thesis has been developed. First, to introduce the idea that diverse IT components provide different capabilities and therefore value although they might increase required costs, the digital options theory described by Sambamurthy et al. (2003) is used. In addition, this idea is backed by the challenge of ambidexterity (see Gibson and Birkinshaw, 2004) requiring both, application landscape standardization and differentiation (Gregory et al., 2015), as well as the resource-based view of the firm (see Wernerfelt, 1984). To guide the development of respective decision support, enterprises are viewed as complex systems (see Buckl et al., 2009). Accordingly, existing literature on complexity has been analyzed and a **complexity framework** describing different prevalent notions of complexity has been developed. This framework has then been used to classify the existing related work—identified by a literature review following the guidelines of Webster and Watson (2002)—which resulted in the observation that both the subjective notion and the dynamic notion of complexity are currently underrepresented in the EAM literature.

In Chapter 3, the two concepts *application landscape complexity* and *application landscape diversity* have been analyzed regarding their antecedents, consequences and management means based on empirical investigations. By means of focus group interviews, typical drivers and consequences of application landscape complexity have been identified to complement the scarce literature in this area. The findings were then complemented by an online survey with 47 participants from industry which revealed that there is a strong interrelation between both concepts in the context of application landscapes. The identified **effects of application landscape diversity** answer the second research question and include, for example, increasing costs as well as technology and vendor independence. By comparing the results of both studies and considering research on complex systems in general, the relationship between application landscape complexity and diversity has been elaborated in detail resulting in both, similarities and differences.

In Chapter 4, a conceptual **framework for application landscape diversity** has been presented. It includes all diversity aspects relevant for the design of application landscapes and is based on ideas and insights developed in other fields including system science and ecology (see Stirling, 2007; Page, 2011). The previously dominant concepts of variety and heterogeneity are thereby complemented by the concepts of balance (for nominal and ordinal data) and disparity. Furthermore, the application of description logics demonstrates a formal way of assessing application landscape diversity by clearly distinguishing between individuals subject to *variation* and concepts subject to *variety*, *balance* and *disparity*. Thus, the conceptual diversity framework provides an answer to the first research question and a shared terminology for future research activities.

In Chapter 5, the core contribution of this thesis has been presented: **quantitative indicators**

to measure application landscape diversity. Therefore, existing complexity and diversity indicators have been identified from literature by following the guidelines of Webster and Watson (2002), complemented by observations of practitioners by following the guidelines of Buckl et al. (2013) for Pattern-based Design Research. Together with several industry experts, benefits and drawbacks of currently existing indicators have been assessed. By following the GQM approach of Basili and Weiss (1984), for each dimension of the described diversity framework concrete indicators have been selected from previous EAM research or designed based on findings from other disciplines. For *variety* and *balance* of unordered EA elements, the indicators proposed by Mocker (2009) and Schuetz et al. (2013) appeared to be suitable. However, for *balance* of ordered EA elements as well as *disparity*, no suitable indicators have been found. Accordingly, the adolescence indicator and a disparity metric have been designed to close this research gap and answer the third research question. In addition, an **implementation of all indicators** has been presented by using the R language together with an architecture description and implementation of a prototypical software solution which answers the fourth research question.

In Chapter 6, the currently underrepresented dynamic notion of complexity in EAM has been addressed by developing a **SD model** explaining organizational behavior regarding application landscape diversity. Thereby, dynamic hypotheses on cause-and-effect relationships have been generated based on respective literature and experience. The CLD emerging from these hypotheses has then been analyzed with regard to its correctness and its ability to foster shared understanding among application landscape designers. The performed expert interviews confirmed the model's validity and revealed its potential to create shared understanding about ongoing dynamics. By modeling relevant phenomena which can be influenced by application landscape designers, complemented by typical diversity management means identified via the practitioner survey, the CLD answers the fifth research question. The CLD has been supplemented with guidelines derived from literature and interviews to foster the creation of organization-specific SD models by practicing enterprise architects.

In Chapter 7, the previously developed artifacts including the adolescence indicator, the disparity indicator, the prototypical software solution and the CLD, are **evaluated via four different case studies** following the recommendations of Benbasat et al. (1987). In the first case study, the disparity indicator has been successfully used to inform a decision regarding the complexity of technical platforms suitable for a business application at a large financial service provider. In the second case study, all four artifacts have been successfully used to assess the complexity and diversity of a mid-sized manufacturing company's application landscape and explain increasing IT costs. In the third case study, the adolescence indicator has been used successfully to identify technical components subject to adolescence within the application landscape of a large insurance company. In the fourth case study, the disparity indicator has been used successfully to support a decision concerning an increase of application server diversity at a large car manufacturing company. Thereby, it has been shown that the artifacts developed and presented in this thesis can be applied in practice.

8.2. Critical reflection

Hevner et al. (2004) outline several quality criteria for design science research including *rigor*

and *relevance*. Beyond the contributions to the knowledge base, some aspects regarding rigor and relevance have been unaddressed in this thesis and have to be considered as limitations.

In Chapter 3, the relationship between application landscape complexity and diversity has been assessed. However, a definite answer to the question how both concepts interrelate could not be derived. Neither literature nor industry expert opinions revealed a clear relationship between both. One reason might be the absence of an agreed-upon definition of both concepts. Therefore, applicability of diversity metrics for complexity quantification might be questioned.

The novel metrics presented in Chapter 5 form the basis of the developed decision support. However, there is no theory available describing in which direction the metric values should be pushed. Therefore, the decision support is limited in so far that it does not provide target values for all kinds of EA elements within the context of every organization. Furthermore, the significance of the described metrics is constrained by the availability of a thorough and up-to-date architectural description. If an organization is not in possession of the required information, the metrics are not applicable. In addition, metrics in general provide only an aggregated view on selected aspects of the system under investigation. Therefore, a single metric should not be used to determine the results of all related decisions.

Considering diversity is always based on a mental model or ontology of a system's observer. As described in Chapter 4, diversity can be regarded among individuals or among concepts describing these individuals. While individuals are different to each other by definition, considering diversity at the concept level has been identified to be more expedient. However, there exists no proven method to define appropriate concepts. Therefore, the significance of the presented metrics is also constrained by the concepts used by the enterprise architect. While this enables their usage for arbitrary EA meta-models, this fact could also be regarded as a limitation. For example, this aspect becomes especially obvious for the disparity metric introduced in Chapter 5.4.4. Although this metric enables enterprise architects to measure disparity for the first time, it requires not only the definition of organization-specific criteria determining the distance between two concepts but also the definition of different characteristics for each type of EA element which should be considered. Accordingly, the significance of the provided metrics relies on the enterprise architect's ability to define an appropriate EA meta-model as well as characteristics important for diversity considerations in his or her individual context.

The SD model presented in Chapter 6 considers only a limited amount of relevant behaviors. Although evaluated with several industry experts, it cannot be excluded that significant behaviors are missing. The same limitation regarding the scope holds true for the performed practitioners survey. The number of 47 participants might not be representative for the whole industry. In addition, participants were all working for German-speaking companies which might limit the ability to transfer the results to other countries.

Although the artifacts presented in this thesis have been successfully evaluated in four case studies, it cannot be assumed, that they are applicable in every context. This especially includes the presented software solution which has not been used in practice productively so far.

8.3. Outlook

The contributions of this thesis provide an innovation in the field of EAM and lay the foundation for future research activities in the field. In the remainder of this chapter, potential future research topics are discussed.

The diversity concept used in this thesis is influenced by the theory of digital options (see Sambamurthy et al., 2003) and previous research on IT standardization (see Boh and Yellin, 2007) as presented in Chapters 2.2 and 2.3. Further related disciplines that could contribute to more sophisticated techniques to guide diversity management are portfolio theory (see Markowitz, 1952), monopoly theory (see Smith and Skinner, 1991) and task-technology fit theory (see Goodhue and Thompson, 1995). Especially the identification of tools or methods to find an optimal degree of diversity in general or within a particular context can be enhanced. In addition, further applications of the developed indicators and the SD model in practice are of interest. Long-term observations of organizations that apply the developed indicators could provide further information on the applicability of the indicators and allow researchers and practitioners to reflect on changing degrees of diversity. Thus, longitudinal studies would open a new research field in which changes of application landscape diversity can be investigated.

The complexity and diversity understanding used within this thesis considers business applications or technical components as the units of analysis. These units might also be subject to an internal complexity or diversity (see Lilienthal, 2009; Partridge and Krzanowski, 1997). Therefore, it is of interest how application landscape complexity and diversity are related to application complexity and diversity. Future research needs to clarify, whether they can be regarded separately or need to be considered simultaneously due to their mutual influence. Of special interest is also, whether the presented indicators are also suitable to quantify diversity within a single software system. Likewise, the relationship between application landscape complexity and business complexity needs to be analyzed in more detail (Schmidt, 2015). Again, the mutual impact of both concepts needs to be understood to guide future design decisions in organizations.

The indicators presented in this thesis allow enterprise architects to quantify relevant aspects of application landscape diversity. The results of the practitioners survey as well as the CLD modeling organizational behavior provide hints how to influence application landscape diversity. With these tools at hand, future research activities can apply statistical methods to assess how application landscape diversity influences other relevant characteristics of an enterprise. These characteristics might include, but are not limited to, adaptability to a changing environment (see e.g. Kandjani et al., 2013), different kinds of IT costs (see Boh and Yellin, 2007) and the complexity of the company's business (see Schmidt, 2015). Therefore, appropriate tool support exceeding the functionality of the presented prototypical solution with, for example, visualization capabilities for different stakeholders and time-line analyses would be desirable.

Bibliography

Kwabena Agyapong-Kodua, Richard H. Weston, and Svetan Ratchev. The integrated use of enterprise and system dynamics modelling techniques in support of business decisions. *Advances in Decision Sciences*, 2012, 2012. ISSN 2090-3359. doi: 10.1155/2012/804324.

Stephan Aier, Stephan Kurpjuweit, Jan Saat, and Robert Winter. Enterprise Architecture Design as an Engineering. *AIS Transactions on Enterprise Systems*, 1(1):36–43, 2009.

Alberto Alesina and Eliana La Ferrara. Ethnic diversity and economic performance. *Journal of Economic Literature*, 43(3):762–800, 2004.

Peter M. Allen. A complex systems approach to learning in adaptive networks. *International Journal of Innovation Management*, 5(2):149–180, 2001. ISSN 1363-9196.

Raphael Amit and Paul J. H. Schoemaker. Strategic assets and organizatioal rent. *Strategische Managementtheorie*, 14(1):33–46, 2012. ISSN 3110803402.

Martha Amram and Nalin Kulatilaka. *Real Options: Managing Strategic Investment in an Uncertain World*. Harvard Business School Press, Cambridge, 1999.

Angela Andal-Ancion, Phillip A. Cartwright, and George S. Yip. The Digital Transformation of Traditional Businesses. *MIT Sloan Management Review*, 44(4):34–41, 2003. ISSN 15329194.

Carl Anderson, Guy Theraulaz, and J-L Deneubourg. Self-assemblages in insect societies. *Insectes Sociaux*, 49(2):99–110, 2002. ISSN 0020-1812.

Pierpaolo Andriani. Diversity, knowledge and complexity theory: some introductory issues. *International Journal of Innovation Management*, 5(02):257–274, 2001. ISSN 1363-9196.

Chris Argyris. *Strategy, change and defensive routines*. Pitman Publishing, 1985. ISBN 0273023292.

W. Brian Arthur. Competing technologies, increasing returns, and lock-in by historical events. *The Economic Journal*, 99(394):116–131, 1989. ISSN 0013-0133.

W. Brian Arthur. Inductive reasoning and bounded rationality. *The American economic review*, pages 406–411, 1994. ISSN 0002-8282.

W. Ross Ashby. Principles of the Self-Organizing Dynamic System. *The Journal of general psychology*, 37(2):125–128, 1947. ISSN 0022-1309.

William Ross Ashby. *An introduction to cybernetics*. Chapman & Hall, London, 1956. ISBN 0-416-68300-2.

Ron Ashkenas. Simplicity-minded management. *Harvard Business Review*, 85(12):101–109, 2007. ISSN 0017-8012.

Nikitas A. Assimakopoulos and Anastasios N. Riggas. Designing a virtual enterprise architecture using structured system dynamics. *Human Systems Management*, 25(1):13–29, 2006. ISSN 0167-2533.

Robert Axelrod. *Structure of decision: The cognitive maps of political elites*. Princeton University Press, 2015. ISBN 1400871956.

Franz Baader, editor. *The description logic handbook: theory, implementation, and applications*. Cambridge University Press, 2003. ISBN 0521781760.

Paul Bachmann. *Die analytische zahlentheorie*. Teubner, 1894.

Michael J. Baker. Writing a literature review. *The Marketing Review*, 1(2):219–247, 2000. ISSN 1469-347X.

Carliss Baldwin, Alan MacCormack, and John Rusnak. Hidden structure: Using network methods to map system architecture. Harvard Business School Working Paper (13-093), 2013.

Sandra Barker and Brenton Fiedler. Developers, Decision Makers, Strategists or Just End-users? Redefining End-User Computing for the 21st Century: A Case Study. *Journal of Organizational and End User Computing (JOEUC)*, 23(2):1–14, 2011. ISSN 1546-2234.

Jay Barney. Firm resources and sustained competitive advantage. *Journal of management*, 17 (1):99–120, 1991. ISSN 0149-2063.

Justus Baron and Julia Schmidt. Technological Standardization, Endogenous Productivity and Transitory Dynamics, 2013.

Victor R. Basili and David M. Weiss. A methodology for collecting valid software engineering data. *IEEE Transactions on Software Engineering*, 10(6):728–738, 1984.

Jörg Becker. Strukturanalogien in Informationsmodellen. In Wolfgang König, editor, *Wirtschaftsinformatik '95*, pages 133–150. Physica-Verlag HD, 1995. ISBN 978-3-642-63388-1.

Stafford Beer. *Cybernetics and Management*. English Universities Press, London, 2nd ed edition, 1967. ISBN 9780340045947.

Danny N. Bellenger, Goldstucker, Kenneth L Bernhardt Jac L, Kenneth L. Bernhardt, and Jac L. Goldstucker. *Qualitative research in marketing*. American Marketing, Chicago, 1976. ISBN 1613112157.

Izak Benbasat, David K. Goldstein, and Melissa Mead. The case research strategy in studies of information systems. *MIS Quarterly*, pages 369–386, 1987.

Mary J. Benner and Michael Tushman. Process management and technological innovation: A longitudinal study of the photography and paint industries. *Administrative science quarterly*, 47(4):676–707, 2002. ISSN 0001-8392.

Daniela Berardi, Diego Calvanese, and Giuseppe de Giacomo. Reasoning on UML class diagrams. *Artificial Intelligence*, 168(1-2):70–118, 2005. ISSN 00043702. doi: 10.1016/j.artint.2005.05.003.

Ludwig von Bertalanffy. *General system theory: Foundations, development, applications*. George Braziller, New York, 1968.

Jesús Bisbal, Deirdre Lawless, Bing Wu, and Jane Grimson. Legacy information systems: Issues and directions. *IEEE Software*, 16(5):103–111, 1999.

Alexander Bogner, Beate Littig, and W. Menz. *Interviewing experts*. Research methods series. Palgrave Macmillan, Basingstoke and New York, 2009. ISBN 0230220193.

Wai Fong Boh and Daniel Yellin. Using enterprise architecture standards in managing information technology. *Journal of Management Information Systems*, 23(3):163–207, 2007. ISSN 0742-1222.

Danail Bonchev and Gregory A. Buck. Quantitative measures of network complexity. In *Complexity in Chemistry, Biology, and Ecology*, pages 191–235. Springer, 2005. ISBN 0387232648.

Charles F. Bond and Linda J. Titus. Social facilitation: a meta-analysis of 241 studies. *Psychological bulletin*, 94(2):265, 1983. ISSN 1939-1455.

Hartmut Bossel. *Systeme, Dynamik, Simulation: Modellbildung, Analyse und Simulation komplexer Systeme*. Books on Demand, Norderstedt, 2004a. ISBN 9783833409844.

Hartmut Bossel. *Systemzoo 3: Wirtschaft, Gesellschaft und Entwicklung*, volume 3 of *Systemzoo*. Books on Demand, Norderstedt, 2004b. ISBN 9783833412417.

Gordon H. Bower and Daniel G. Morrow. Mental models in narrative comprehension. *Science*, 247(4938):44–48, 1990. ISSN 0036-8075.

G. A. Britton and J. Parker. An explication of the viable system model for project management. *Systems Practice*, 6(1):21–51, 1993. ISSN 0894-9859.

John Brocklesby and Stephen Cummings. Designing a viable organization structure. *Long Range Planning*, 29(1):49–57, 1996. ISSN 0024-6301.

Bernd Bruegge and Allen H. Dutoit. *Object-oriented software engineering: Using UML, patterns, and Java*. Prentice Hall, Boston, 3rd ed. edition, 2010. ISBN 0136061257.

G. Brys, Mia Hubert, and A. Struyf. A robust measure of skewness. *Journal of Computational and Graphical Statistics*, 13(4):996–1017, 2004.

Sabine Buckl, Alexander M. Ernst, Josef Lankes, Kathrin Schneider, and Christian M. Schweda. A pattern based approach for constructing enterprise architecture management information models. *Wirtschaftinformatik Proceedings 2007*, page 65, 2007.

Sabine Buckl, Alexander M. Ernst, Josef Lankes, and Florian Matthes. Enterprise Architecture Management Pattern Catalog, 2008.

Sabine Buckl, Florian Matthes, and Christian M. Schweda. A Viable System Perspective on Enterprise Architecture Management. In *Proceedings of the International Conference on Systems, Man and Cybernetics*. IEEE, 2009. ISBN 978-1-4244-2794-9.

Sabine Buckl, Thomas Dierl, Florian Matthes, and Christian M. Schweda. Building Blocks for Enterprise Architecture Management Solutions. In *Practice-driven research on enterprise transformation*, pages 17–46. Springer, 2010. ISBN 3642167691.

Sabine Buckl, Florian Matthes, Alexander W. Schneider, and Christian M. Schweda. Pattern-based Design Research–An Iterative Research Method Balancing Rigor and Relevance. In Jan vom Brocke, Riitta Hekkala, Sudha Ram, and Matti Rossi, editors, *Design Science at the Intersection of Physical and Virtual Design*, pages 73–87. Springer, Berlin, Heidelberg, 2013. ISBN 3642388264.

Sabine M. Buckl. *Developing organization-specific enterprise architecture management functions using a method base*. PhD thesis, München, Technische Universität München, Diss., 2011, 2011.

Mario Augusto Bunge. *A world of systems*, volume 4 of *Treatise on basic philosophy*. D. Reidel Publishing, Boston, 1979.

Peter Buxmann, Tim Weitzel, Falk von Westarp, and Wolfgang König. The standardization problem: an economic analysis of standards in information systems. In K. Jakobs and Williams R., editors, *Proceedings of the 1st IEEE conference on standardisation and innovation in information technology*, 1999.

Bobby J. Calder. Focus groups and the nature of qualitative marketing research. *Journal of Marketing research*, 47(4):353–364, 1977. ISSN 0022-2437.

Gianluigi Caldiera, Victor R. Basili, and Dieter Rombach. Goal question metric paradigm. *Encyclopedia of Software Engineering*, 1:528–532, 1994.

Donald Thomas Campbell, Julian C. Stanley, and Nathaniel Lees Gage. Experimental and quasi-experimental designs for research, 1963.

Kevin Campbell and Antonio Mínguez-Vera. Gender Diversity in the Boardroom and Firm Financial Performance. *Journal of Business Ethics*, 83(3):435–451, 2008. ISSN 0167-4544. doi: 10.1007/s10551-007-9630-y.

Capgemini. IT-Trends 2014: IT-Kompetenz im Management steigt, 2014.

Rudolf Carnap. *The logical structure of the world -: And pseudoproblems in philosophy*. Open Court classics. Open Court, Chicago and La Salle, Ill., 2003. ISBN 9780812695236.

Yolande E. Chan, Sid L. Huff, Donald W. Barclay, and Duncan G. Copeland. Business strategic orientation, information systems strategic orientation, and strategic alignment. *Information Systems Research*, 8(2):125–150, 1997.

Yolande E. Chan, Rajiv Sabherwal, and Jason Bennett Thatcher. Antecedents and outcomes of strategic IS alignment: an empirical investigation. *Engineering Management, IEEE Transactions on*, 53(1):27–47, 2006. ISSN 0018-9391.

Surajit Chaudhuri and Umeshwar Dayal. An overview of data warehousing and OLAP technology. *ACM SIGMOD Record*, 26(1):65–74, 1997. ISSN 01635808. doi: 10.1145/248603.248616.

Peter Checkland. From optimizing to learning: A development of systems thinking for the 1990s. *Journal of the Operational Research Society*, pages 757–767, 1985. ISSN 0160-5682.

Patricia W. Cheng and Keith J. Holyoak. Pragmatic reasoning schemas. *Cognitive psychology*, 17(4):391–416, 1985. ISSN 0010-0285.

Anne Cleven, Philipp Gubler, and Kai M. Hüner. Design alternatives for the evaluation of design science research artifacts. In *Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology*. ACM, 2009. ISBN 1605584088.

Edgar F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, 1970. ISSN 00010782.

Wesley M. Cohen and Daniel A. Levinthal. Absorptive capacity: a new perspective on learning and innovation. *Administrative science quarterly*, pages 128–152, 1990. ISSN 0001-8392.

Roger C. Conant and W. Ross Ashby. Every good regulator of a system must be a model of that system†. *International journal of systems science*, 1(2):89–97, 1970. ISSN 0020-7721.

Matthew Cook. Universality in elementary cellular automata. *Complex Systems*, 15(1):1–40, 2004. ISSN 0891-2513.

Harris M. Cooper. Organizing knowledge syntheses: A taxonomy of literature reviews. *Knowledge in Society*, 1(1):104–126, 1988. ISSN 0897-1986.

Paul A. David. Clio and the Economics of QWERTY. *The American economic review*, pages 332–337, 1985. ISSN 0002-8282.

Kenyon B. De Greene. *Sociotechnical systems: factors in analysis, design, and management*. Prentice-Hall, Englewood Cliffs, N.J., 1973. ISBN 9780138215538.

Tom DeMarco. *Controlling software projects: Management, measurement & estimation*. Yourdon Press, New York, NY, 1982. ISBN 9780131717114.

Gernot Dern and R. Jung. IT-Architektur-Governance auf Basis von Kennzahlen zur Komplexitätsmessung. *Zeitschrift für Controlling*, 21(12):669–672, 2009.

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. ISSN 0742-1222.

Rohit Deshpande. Paradigms Lost: On Theory and Method in Research in Marketing. *Journal of Marketing*, 47(4):101, 1983. ISSN 00222429. doi: 10.2307/1251403.

Michael Diehl and Wolfgang Stroebe. Productivity loss in brainstorming groups: Toward the solution of a riddle. *Journal of personality and social psychology*, 53(3):497, 1987. ISSN 1939-1315.

Edsger Wybe Dijkstra. *Selected writings on computing: A personal perspective*. Texts and monographs in computer science. Springer-Verlag, New York, 1982. ISBN 9783540906520.

U.S.A DoD. Department of Defense Architecture Framework (DoDAF), 2009. URL `http://dodcio.defense.gov/Portals/0/Documents/DODAF/DoDAF_v2-02_web.pdf`.

Niles Eldredge. *Systematics, ecology, and the biodiversity crisis*. Columbia University Press, 1992. ISBN 0231075286.

Fred E. Emery. Characteristics of socio-technical systems: The emergence of a new paradigm of work. *ANU/CCE: Canberra*, 1959.

Alexander M. Ernst. Enterprise Architecture Management Patterns. In Joseph Yoder and Ademar Aguiar, editors, *Proceedings of the 15th Conference on Pattern Languages of Programs*. ACM, 2008. ISBN 1605581518.

Brian Everitt. *The Cambridge Dictionary of Statistics*. Cambridge University Press, Cambridge, UK and New York, 1998. ISBN 9780521593465.

Terje Fallmyr and Bendik Bygstad. Enterprise Architecture Practice and Organizational Agility: An Exploratory Study. In *47th Hawaii International Conference on System Sciences (HICSS)*, pages 3788–3797, 2014. doi: 10.1109/HICSS.2014.471.

Joseph Farrell and Paul Klemperer. Coordination and lock-in: Competition with switching costs and network effects. *Handbook of industrial organization*, 3:1967–2072, 2007. ISSN 1573-448X.

Matthias Farwick, Berthold Agreiter, Ruth Breu, Steffen Ryll, Karsten Voges, and Inge Hanschke. Requirements for automated enterprise architecture model maintenance. In Runtong Zhang, José Cordeiro, Xuewei Li, Zhenji Zhang, and Juliang Zhang, editors, *Proceedings of the 13th international conference on enterprise information systems*, pages 325–337. SciTePress, 2011. ISBN 9789898425560.

Norman E. Fenton and Shari Lawrence Pfleeger. *Software metrics: A rigorous and practical approach*. PWS Pub., Boston, 2nd edition, 1997. ISBN 9780534954253.

Walter J. Ferrier, Ken G. Smith, and Curtis M. Grimm. The role of competitive action in market share erosion and industry dethronement: A study of industry leaders and challengers. *Academy of management journal*, 42(4):372–388, 1999. ISSN 0001-4273.

Robert G. Fichman. Real options and IT platform adoption: Implications for theory and practice. *Information Systems Research*, 15(2):132–154, 2004.

Michael O. Finkelstein and Richard M. Friedberg. The application of an entropy theory of concentration to the Clayton Act. *Yale Law Journal*, pages 677–717, 1967. ISSN 0044-0094.

Guido Fioretti. A subjective measure of complexity. *Advances in Complex Systems*, 2(04): 349–370, 1999. ISSN 0219-5259.

M. Flückiger and M. Rauterberg. Komplexität und Messung von Komplexität, 1995.

David N. Ford and John D. Sterman. Expert knowledge elicitation to improve formal and mental models. *System Dynamics Review*, 14(4):309–340, 1998. ISSN 1099-1727.

Jay Wright Forrester. *Industrial dynamics*. MIT press Cambridge, MA, 1961.

Martin Fowler and Pramod J. Sadalage. *NoSQL distilled*. Addison-Wesley, Boston, Mass. and London, 2012. ISBN 0321826620.

Ulrich Frank. Multi-perspective enterprise modeling (MEMO) -conceptual framework and modeling languages. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS)*, pages 1258–1267. IEEE, 2002. ISBN 0769514359.

Ulrich Frank. Towards a pluralistic conception of research methods in information systems research, 2006.

Charles R. Franz and Daniel Robey. An investigation of user-led system design: rational and political perspectives. *Communications of the ACM*, 27(12):1202–1209, 1984. ISSN 00010782.

Michael Frese. A theory of control and complexity: Implications for software design and integration of computer system into the work place. In M. Frese, E. Ulich, and W. Dzida, editors, *Psychological Issues of Human Computer Interaction in the Work Place*. Elsevier Science Publishers B.V., 1987.

Daniel Fürstenau and Natalia Kliewer. Exploring Enterprise Transformation from a Path Dependence Perspective: A Recycling Case and Conceptual Model. In O. Thomas and F. Teutenberg, editors, *Proceedings der 12. Internationalen Tagung Wirtschaftsinformatik (WI 2015)*, pages 363–377, 2015.

Hans-Georg Gadamer. Hermeneutics and Social Science. *Philosophy & Social Criticism*, 2(4): 307–316, 1975. ISSN 0191-4537.

Robert D. Galliers. Strategizing for Agility: Confronting Information. In Kevin C. Desouza, editor, *Agile Information Systems*, pages 1–15. Routledge, 2006. ISBN 978-0-7506-8235-0.

Martin Gardner. Mathematical games: The fantastic combinations of John Conway's new solitaire game "life". *Scientific American*, 223(4):120–123, 1970.

Abd Ghani, Abdul Azim, Tieng Wei Koh, Geoffrey Muchiri Muketha, and Pei Wen Wong. Complexity metrics for measuring the understandability and maintainability of business process models using goal-question-metric (GQM). *International Journal of Computer Science and Network Security*, 8(5):219–225, 2008. ISSN 1738-7906.

Arnab Ghosh, Prashant Kumar Gajar, and Shashikant Rai. Bring your own device (BYOD): Security risks and mitigating strategies. *Journal of Global Research in Computer Science*, 4 (4):62–70, 2013. ISSN 2229-371X.

Cristina B. Gibson and Julian Birkinshaw. The antecedents, consequences, and mediating role of organizational ambidexterity. *Academy of management journal*, 47(2):209–226, 2004. ISSN 0001-4273.

Corrado Gini. Variabilità e mutabilità. *Reprinted in Memorie di metodologica statistica (Ed. Pizetti E, Salvemini, T). Rome: Libreria Eredi Virgilio Veschi*, 1, 1912.

John Goetze and Anders Jenssen-Waud, editors. *Beyond alignment: Applying systems thinking in architecting enterprises*. College Publications, 2013. ISBN 9781848901162.

Joseph A. Goguen and Charlotte Linde. Techniques for requirements elicitation. *Proc. RE'93 - First IEEE Symposium on Requirements Engineering*, pages 152–164, 1993.

Jeffrey Goldstein. Emergence as a construct: History and issues. *Emergence*, 1(1):49–72, 1999. ISSN 1521-3250.

Arash Golnam, Ann van Ackere, and Alain Wegmann. Integrating system dynamics and enterprise modeling to address dynamic and structural complexities of choice situations. In Tae-Hoon Moon, editor, *Proceedings of The 28th International Conference of The System Dynamics Society*. Wiley, 2010. ISBN 978-1-935056-05-8.

Dale L. Goodhue and Ronald L. Thompson. Task-technology fit and individual performance. *MIS Quarterly*, 19(2):213–236, 1995.

Ove Granstrand and Christer Oskarsson. Technology diversification in "MUL-TECH" corporations. *IEEE transactions on Engineering Management*, 41(4):355–364, 1994. ISSN 0018-9391.

Jim Gray. The transaction concept: Virtues and limitations. In *Proceedings of the Seventh International Conference Very Large Databases*. Tandem Computers Incorporated, 1981.

Danny Greefhorst and Henderik Alex Proper. *Architecture Principles*. Springer, Berlin, Heidelberg, 2011. ISBN 978-3-642-20279-7.

Robert Gregory, Mark Keil, Jan Muntermann, and Magnus Mähring. Paradoxes and the Nature of Ambidexterity in IT Transformation Programs. *Information Systems Research*, 26(1), 2015.

Richard A. Groeneveld and Glen Meeden. Measuring Skewness and Kurtosis. *The Statistician*, 33(4):391–399, 1984. ISSN 00390526.

Andreas Györy, Anne Cleven, Falk Uebernickel, and Walter Brenner. Exploring the shadows: IT governance approaches to user-driven innovation. *Proceedings of the 20th European Conference on Information Systems, Barcelona, Spain*, 2012.

Jürgen Habermas. *Theorie des kommunikativen Handelns: Vol 1: Handlungsrationalikit und gesellschaftliche Rationalisierung*. Suhrkamp, Frankfurt/M, 1981.

Marshall Hall and Nicolaus Tideman. Measures of concentration. *Journal of the American Statistical Association*, 62(317):162–168, 1967. ISSN 0162-1459.

Andreas Handl. *Maßzahlen zur Klassifizierung von Verteilungen bei der Konstruktion adaptiver verteilungsfreier Tests im unverbundenen Zweistichproben-Problem*. PhD thesis, Freie Universität Berlin, Berlin, 1985.

Inge Hanschke. *Strategisches Management der IT-Landschaft: Ein praktischer Leitfaden für das Enterprise-Architecture-Management*. Hanser, München, 2013. ISBN 3446435093.

Inge Hanschke. *Lean IT-Management–einfach und effektiv: Der Erfolgsfaktor für ein wirksames IT-Management*. Carl Hanser Verlag GmbH Co KG, 2014. ISBN 3446441999.

Ole Hanseth, Eric Monteiro, and Morten Hatling. Developing information infrastructure: The tension between standardization and flexibility. *Science, technology & human values*, 21(4): 407–426, 1996. ISSN 0162-2439.

James Harrington. In my opinion. *CIO*, 12(23):19, 1999.

David A. Harrison and Katherine J. Klein. What's the difference? Diversity constructs as separation, variety, or disparity in organizations. *Academy of Management Review*, 32(4): 1199–1228, 2007. ISSN 03637425.

Matheus Hauder, Sascha Roth, Christopher Schulz, and Florian Matthes. Current Tool Support for Metrics in Enterprise Architecture Management. In Günter Büren, Reiner R. Dumke, Christof Ebert, Jürgen Münch, and Manfred Seufert, editors, *DASMA Software Metrik Kongress (Metrikon 2013)*, 2013a. ISBN 978-3-8440-2350-3.

Matheus Hauder, Sascha Roth, Christopher Schulz, and Florian Matthes. An Examination Of Organizational Factors Influencing Enterprise Architecture Management Challenges. In *Proceedings of the 21st European Conference on Information Systems*, 2013b.

Orris C. Herfindahl. *Concentration in the US steel industry*. Columbia University, 1950.

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

M. O. Hill. Diversity and Evenness: A Unifying Notation and Its Consequences. *Ecology*, 54(2): 427, 1973. ISSN 0012-9658. doi: 10.2307/1934352.

John H. Holland. *Hidden order: How adaptation builds complexity*. Helix books. Addison-Wesley, Reading, Mass., 1995. ISBN 9780201442304.

Jack Homer and Rogelio Oliva. Maps and models in system dynamics: a response to Coyle. *System Dynamics Review*, 17(4):347–355, 2001. ISSN 1099-1727. doi: 10.1002/sdr.224.

Janos Horvath. Suggestion for a comprehensive measure of concentration. *Southern Economic Journal*, pages 446–452, 1970. ISSN 0038-4038.

Liu Hui, Bai Dianyi, and Liu Jin. A study on the mechanism of relationship among intellectual property rights, technical innovation and standardization. In *6th International Conference on Information Management, Innovation Management and Industrial Engineering (ICIII)*, pages 593–597. IEEE, 2013. ISBN 978-1-4799-3985-5. doi: 10.1109/ICIII.2013.6703656.

Jez Humble and David Farley. *Continuous delivery: reliable software releases through build, test, and deployment automation*. Pearson Education, 2010. ISBN 0321670221.

International Organization for Standardization. ISO/IEC 42010:2007 Systems and Software Engineering - Recommended Practice for Architectural Description of Software-Intensive Systems (ISO/IEC 42010 IEEE Std 1471-2000), 2007.

Scott G. Isaksen and Donald J. Treffinger. Creative problem solving. *The Basic Course. New York: Bearly Limited*, 1985.

Paul Jaccard. The Distribution of the Flora in the Alpine Zone. *New Phytologist*, 11(2):37–50, 1912. ISSN 0028-646X.

Kai Jakobs. *Advanced Topics in Information Technology Standards and Standardization Research, Volume 1*. IGI Global, 2005. ISBN 1591409403.

Marijn Janssen and George Kuk. A Complex Adaptive System Perspective of Enterprise Architecture in Electronic Government. In *39th Hawaii International Conference on System Sciences (HICSS)*, 2006.

Erica Jen. *Robust design: Repertoire of biological, ecological, and engineering case studies*. Santa Fe Institute studies in the sciences of complexity. Oxford University Press, New York, 2005. ISBN 0195165322.

Neil F. Johnson. *Two's Company, Three is Complexity: A simple guide to the science of all sciences*. Oneworld, Oxford, 2007. ISBN 1851684883.

David H. Jonassen. Instructional design models for well-structured and III-structured problem-solving learning outcomes. *Educational Technology Research and Development*, 45(1):65–94, 1997. ISSN 1042-1629.

Gareth R. Jones. *Organizational theory, design, and change*. Pearson, 2010. ISBN 0138157111.

Alicia Juarrero. Dynamics in action: intentional behavior as a complex system. *Emergence*, 2 (2):24–57, 2000. ISSN 1521-3250.

S. H. Kaisler, F. Armour, and M. Valivullah. Enterprise Architecting: Critical Problems. In *38th Annual Hawaii International Conference on System Sciences*, 2005. doi: 10.1109/HICSS.2005.241.

Hadi Kandjani, Peter Bernus, and Lian Wen. Enterprise Architecture Cybernetics for Complex Global Software Development: Reducing the Complexity of Global Software Development Using Extended Axiomatic Design Theory. In *7th IEEE International Conference on Global Software Engineering (ICGSE)*, pages 169–173, 2012. doi: 10.1109/ICGSE.2012.19.

Hadi Kandjani, Peter Bernus, and Sue Nielsen. Enterprise Architecture Cybernetics and the Edge of Chaos: Sustaining Enterprises as Complex Systems in Complex Business Environments. In *46th Hawaii International Conference on System Sciences (HICSS)*, pages 3858–3867, 2013. doi: 10.1109/HICSS.2013.199.

Steven J. Karau and Kipling D. Williams. Social loafing: A meta-analytic review and theoretical integration. *Journal of personality and social psychology*, 65(4):681, 1993. ISSN 1939-1315.

Panos Kardasis and Peri Loucopoulos. Aligning legacy information systems to business processes. In *Advanced Information Systems Engineering*, volume 1413 of *Lecture Notes in Computer Science*, pages 25–39. Springer, 1998.

Stuart A. Kauffman. *The origins of order: Self-organization and selection in evolution*. Oxford University Press, New York, 1993. ISBN 9780195079517.

Stuart A. Kauffman. *At home in the universe: The search for laws of self-organization and complexity*. Oxford University Press, New York, 1995. ISBN 9780195111309.

Wolfgang Keller. *IT-Unternehmensarchitektur: Von der Geschäftsstrategie zur optimalen IT-Unterstützung*. dpunkt.verlag, Heidelberg, 2 edition, 2012. ISBN 3898647684.

Alfons Kemper and Thomas Neumann. HyPer: A hybrid OLTP&OLAP main memory database system based on virtual memory snapshots. In *27th International Conference on Data Engineering (ICDE)*. IEEE Operations Center, 2011. ISBN 1424489598.

William J. Kettinger, Donald A. Marchand, and Joshua M. Davis. Designing enterprise it architectures to optimize flexibility and standardization in global business. *MIS Quarterly Executive*, 9(2):95–113, 2010.

Jan H. Keuntje and Reinhard Barkow. *Enterprise-architecture-Management in der Praxis: Wandel, Komplexität und IT-Kosten im Unternehmen beherrschen*. Symposion, Düsseldorf, 1 edition, 2010. ISBN 3939707708.

Young-Gul Kim and Gordon C. Everest. Building an IS architecture: Collective wisdom from the field. *Information & Management*, 26(1):1–11, 1994. ISSN 0378-7206.

Alan P. Kirman and Nicolaas J. Vriend. Evolving market structure: An ACE model of price dispersion and loyalty. *Journal of Economic Dynamics and Control*, 25(3):459–502, 2001. ISSN 0165-1889.

Ingo Klein. Systematik der Schiefemessung für ordinalskalierte Merkmale, 1999.

Andrei N. Kolmogorov. On tables of random numbers. *Sankhya: The Indian Journal of Statistics, Series A*, pages 369–376, 1963.

Svyatoslav Kotusev, Mohini Singh, and Ian Storey. Consolidating Enterprise Architecture Management Research. In *48th Hawaii International Conference on System Sciences (HICSS)*, 2015.

David C. Krakauer. Robustness in Biological Systems: a provisional taxonomy. In *Complex systems science in biomedicine*, pages 183–205. Springer, 2006. ISBN 0387302417.

Thomas S. Kuhn. *The structure of scientific revolutions*. University of Chicago press, 1962. ISBN 0226458148.

Narayanan Kumbakara. Managed IT services: the role of IT standards. *Information Management & Computer Security*, 16(4):336–359, 2008. ISSN 0968-5227.

James Ladyman, James Lambert, and Karoline Wiesner. What is a complex system? *European Journal for Philosophy of Science*, 3(1):33–67, 2013. ISSN 1879-4912.

Robert Lagerström, Carliss Baldwin, Alan MacCormack, and Stephan Aier. Visualizing and Measuring Enterprise Application Architecture: An Exploratory Telecom Case. In *47th Hawaii International Conference on System Sciences (HICSS)*, 2014.

Rolf Landauer. A simple measure of complexity. *Nature*, 336:306–307, 1988. ISSN 0028-0836.

DavidC Lane. With a little help from our friends: how third generation system dynamics and the problem structuring techniques of'soft'OR can learn from each other. *City University Business School Discussion Paper ITM/93/DCL2.(A shortened version of this paper is available in (1993) System Dynamics*, pages 235–244, 1993.

Joscf Lankes, Florian Matthes, and André Wittenburg. Softwarekartographie: Systematische Darstellung von Anwendungslandschaften. In O. K. Ferstl, E. J. Sinz, S. Eckert, and T. Isselhorst, editors, *7. Internationale Tagung Wirtschaftsinformatik*, pages 1443–1462. Physica-Verlag, 2005. ISBN 3790815748.

Marc Lankhorst, editor. *Enterprise Architecture at Work*. Springer, 2009. ISBN 3642296505.

James Lapalme and Donald W. de Guerre. An Open Socio-Technical Systems Approach to Enterprise Architecture. In John Goetze and Anders Jenssen-Waud, editors, *Beyond alignment*. College Publications, 2013. ISBN 9781848901162.

Daniel A. Levinthal and James G. March. The myopia of learning. *Strategic Management Journal*, 14(S2):95–112, 1993. ISSN 1097-0266.

Carola Lilienthal. Architectural complexity of large-scale software systems. In *13th European Conference on Software Maintenance and Reengineering*. IEEE, 2009. ISBN 0769535895.

Paul Lorenzen. *Konstruktive Wissenschaftstheorie*, volume 93 of *Suhrkamp Taschenbuch Wissenschaft*. Suhrkamp, Frankfurt am Main, 1974. ISBN 9783518076934.

Jerry Luftman. Assessing business-IT alignment maturity. *Communications of the Association for Information Systems*, 4, 2000. ISSN 1529-3181.

Niklas Luhmann. *Soziale systeme*. Suhrkamp, Frankfurt am Main, 1984. ISBN 3518576844.

Luis Felipe Luna-Reyes and Deborah Lines Andersen. Collecting and analyzing qualitative data for system dynamics: methods and models. *System Dynamics Review*, 19(4):271–296, 2003. ISSN 1099-1727.

Robert H. MacArthur. Patterns of species diversity. *Biological reviews*, 40(4):510–533, 1965. ISSN 1469-185X.

Robert H. MacArthur and John W. MacArthur. On Bird Species Diversity. *Ecology*, 42(3):594, 1961. ISSN 0012-9658. doi: 10.2307/1932254.

Klaus Mainzer. *Thinking in complexity: the computational dynamics of matter, mind, and mankind*. Springer Science & Business Media, 2007. ISBN 3540722289.

James G. March. Exploration and exploitation in organizational learning. *Organization Science*, 2(1):71–87, 1991. ISSN 1047-7039.

Salvatore T. March and Gerald F. Smith. Design and Natural Science Research on Information Technology. *Decision Support Systems*, 15(4):251–266, 1995. ISSN 0167-9236.

Harry Markowitz. Portfolio selection*. *The Journal of Finance*, 7(1):77–91, 1952. ISSN 1540-6261.

Nathaniel J. Mass. *Stock and flow variables and the dynamics of supply and demand*. Alfred P. Sloan School of Management. System Dynamics Group, 1977.

Florian Matthes, Sabine Buckl, Jana Leitel, and Christian M. Schweda. *Enterprise architecture management tool survey 2008*. Techn. Univ. München, 2008. ISBN 3000245200.

Florian Matthes, Christian Neubert, and Alexander Steinhoff. Hybrid Wikis: Empowering Users to Collaboratively Structure Information. In *6th International Conference on Software and Data Technologies (ICSOFT)*, pages 250–259, 2011.

Florian Matthes, Ivan Monahov, Alexander W. Schneider, and Christopher Schulz. Towards a unified and configurable structure for EA management KPIs. In *Trends in Enterprise Architecture Research and Practice-Driven Research on Enterprise Transformation*, pages 284–299. Springer, 2012. ISBN 3642341624.

Florian Matthes, Christian Neubert, and Alexander W. Schneider. Fostering Collaborative and Integrated Model and Meta-Model Development with Hybrid Wikis. *Journal of Enterprise Modelling and Information Systems Architectures*, 8(1), 2013.

Florian Matthes, Alexander W. Schneider, Matheus Hauder, and Pouya Aleatrati. EAM Pattern Catalog v2: (to appear), 2015.

Robert M. May. Taxonomy as destiny. *Nature*, 347(6289):129–130, 1990. ISSN 0028-0836.

Yasmin Merali and Bill McKelvey. Using Complexity Science to effect a paradigm shift in Information Systems for the 21st century. *Journal of Information Technologz*, 21:211–215, 2006.

Stanley Milgram. The small world problem. *Psychology today*, 2(1):60–67, 1967.

Melanie Mitchell. *Complexity: A guided tour*. Oxford University Press, Oxford and New York, 2011. ISBN 0199798109.

Martin Mocker. What is Complex about 273 Applications? Untangling Application Architecture Complexity in a case of European Investment Banking. In *42nd Hawaii International Conference on System Sciences (HICSS)*, 2009. ISBN 0769534503.

Martin Mocker and Jeanne W. Ross. USAA: Capturing Value from Complexity, 2013.

U. K. MoD. Ministry of defence architecture framework, 2008. URL `https://www.gov.uk/mod-architecture-framework`.

Wolfgang A. Molnar and Janne J. Korhonen. Research paradigms and topics in Enterprise Engineering analysis of recent conferences and workshops. *Eighth International Conference on Research Challenges in Information Science (RCIS)*, 2014. doi: 10.1109/RCIS.2014.6861071.

Benoit Morel and Rangaraj Ramanujam. Through the looking glass of complexity: The dynamics of organizations as adaptive and evolving systems. *Organization Science*, 10(3):278–293, 1999. ISSN 1047-7039.

Christoph Moser, Stefan Junginger, Matthias Brückmann, and Klaus-Manfred Schöne. Some Process Patterns for Enterprise Architecture Management. In *Software Engineering (Workshops)*, pages 19–30, 2009.

Thomas Mueller, Sven Dittes, Frederik Ahlemann, Nils Urbach, and Stefan Smolnik. Because Everybody is Different: Towards Understanding the Acceptance of Organizational IT Standards. In *48th Hawaii International Conference on System Sciences (HICSS)*, 2015.

Stephan Murer, Carl Worms, and Frank J. Furrer. Managed evolution. *Informatik-Spektrum*, 31(6):537–547, 2008. ISSN 0170-6012.

David G. Myers. Group Polarization. In John M. Levine and Michael A. Hogg, editors, *Encyclopedia of group processes and intergroup relations*, pages 361–365. Sage Publications, 2009. ISBN 1452261504.

R. Nargundkar. *Marketing Research-Text & Cases 2E.* Tata McGraw-Hill, 2003. ISBN 9780070528055.

John von Neumann and Oskar Morgenstern. *Theory of games and economic behavior.* Princeton University Press, Princeton, 1953.

Theodora N. Ngosi and John O. Jenkins. Software standards: an information requirements framework. *Journal of Information Technology*, 8(2):82–91, 1993. ISSN 0268-3962.

Klaus D. Niemann. *Von der Unternehmensarchitektur zur IT-Governance: Bausteine für ein wirksames IT-Management.* Edition CIO. Vieweg, Wiesbaden, 1. edition, 2005. ISBN 9783528058562.

Eetu Niemi. Enterprise architecture benefits: Perceptions from literature and practice. In Eetu Niemi, Tanja Ylimäki, and Niina Hämäläinen, editors, *Evaluation of enterprise and software architectures: critical issues, metrics and practices.* University of Jyväskylä, 2008.

Eetu Niemi and Samuli Pekkola. Enterprise Architecture Quality Attributes: A Case Study. In *46th Hawaii International Conference on System Sciences (HICSS)*, pages 3878–3887, 2013. doi: 10.1109/HICSS.2013.201.

Wonseok Oh and Alain Pinsonneault. On the assessment of the strategic value of information technologies: conceptual and analytical approaches. *MIS Quarterly*, pages 239–265, 2007.

Hannu Oja. On location, scale, skewness and kurtosis of univariate distributions. *Scandinavian Journal of Statistics*, pages 154–168, 1981. ISSN 0303-6898.

Maureen A. O'Leary, Jonathan I. Bloch, John J. Flynn, Timothy J. Gaudin, Andres Giallombardo, Norberto P. Giannini, Suzann L. Goldberg, Brian P. Kraatz, Zhe-Xi Luo, and Jin Meng. The placental mammal ancestor and the post–K-Pg radiation of placentals. *Science*, 339(6120):662–667, 2013. ISSN 0036-8075.

Wanda J. Orlikowski and Stephen R. Barley. Technology and institutions: What can research on information technology and research on organizations learn from each other? *MIS Quarterly*, 25(2):145–165, 2001.

Günther Ossimitz. *Entwicklung systemischen Denkens: theoretische Konzepte und empirische Untersuchungen*. Profil, 2000.

Scott E. Page. *Diversity and complexity*. Primers in complex systems. Princeton University Press, Princeton, N.J, 2011. ISBN 1400835143.

Vilfredo Pareto. *Manuale di economia politica*. Societa Editrice, 1906.

Hang Yeop Park, Sung Hwa Jung, Young-Joong Lee, and Ki Cheol Jang. The Effect of Improving IT Standard in IT Governance. In *International Conference on Computational Intelligence for Modelling, Control and Automation*. IEEE, 2006. ISBN 0769527310.

Derek Partridge and W. Krzanowski. Software diversity: practical statistics for its measurement and exploitation. *Information and Software Technology*, 39(10):707–717, 1997. ISSN 0950-5849.

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. ISSN 0742-1222.

Edith Tilton Penrose. *The Theory of the Growth of the Firm*. Wiley, New York, 1959. ISBN 0198289774.

Evelyn C. Pielou. An introduction to mathematical ecology. *An introduction to mathematical ecology*, 1969.

Geert Poels and Guido Dedene. Distance-based software measurement: necessary and sufficient properties for software measures. *Information and Software Technology*, 42(1):35–46, 2000. ISSN 0950-5849.

Henri Poincaré. *Science and method*. Dover Publications, Mineola, N.Y, dover ed edition, 2003. ISBN 9780486432694.

Karl Popper and Hubert Kiesewetter, editors. *Die offene Gesellschaft und ihre Feinde I*. Mohr Siebeck, Tübingen, 8 edition, 2003. ISBN 3-16-148068-6.

David S. Preston and Elena Karahanna. Antecedents of IS strategic alignment: a nomological network. *Information Systems Research*, 20(2):159–179, 2009.

Thomas Puschmann and Rainer Alt. Enterprise application integration systems and architecture-the case of the Robert Bosch Group. *Journal of Enterprise Information Management*, 17(2):105–116, 2004. ISSN 1741-0398.

R Core Team. R: A Language and Environment for Statistical Computing, 2014. URL `http://www.R-project.org/`.

Murali Ramakrishnan. Software release management. *Bell Labs Technical Journal*, 9(1):205–210, 2004. ISSN 10897089. doi: 10.1002/bltj.20015.

S. Rashid, Tariq Masood, and R. H. Weston. Unified modelling in support of organization design and change. *Journal of Engineering Manufacture*, 223(8):1055–1079, 2009.

Eberhardt Rechtin. *Systems architecting of organizations: Why eagles can't swim.* CRC Press, 1999. ISBN 0849381401.

Christopher Rentrop and Stephan Zimmermann. Shadow IT. *Management and Control of Unofficial IT. ICDS*, pages 98–102, 2012.

Thomas Reschenhofer, Ivan Monahov, and Florian Matthes. Type-Safety in EA Model Analysis. In *18th International Enterprise Distributed Object Computing Conference Workshops and Demonstrations (EDOCW)*, pages 87–94. IEEE, 2014.

Gary L. Richardson, Brad M. Jackson, and Gary W. Dickson. A principles-based enterprise architecture: Lessons from Texaco and Star Enterprise. *MIS Quarterly*, pages 385–403, 1990.

George P. Richardson. *Feedback thought in social science and systems theory.* System dynamics series. Pegasus Communications, Waltham, MA, 1999. ISBN 9781883823467.

Albrecht Richen and Ansgar Steinhorst. Standardization or harmonization? you need both. *European Health Informatics (November 2005)*, page 5, 2005.

Fritz Jules Roethlisberger. *The Elusive Phenomena.* Harvard Business School, Division of Research, Bosten, Massachusetts, 1977.

Robert Rosen. Complexity as a System Property. *International Journal of General Systems*, 3 (4):227–232, 1977. ISSN 0308-1079. doi: 10.1080/03081077708934768.

Jeanne W. Ross, Peter Weill, and David C. Robertson. *Enterprise Architecture as Strategy: Creating a Foundation for Business Execution.* Harvard Business Press, 2006. ISBN 1591398398.

Jennifer Rowley and Frances Slack. Conducting a literature review. *Management Research News*, 27(6):31–39, 2004. ISSN 0140-9174.

Jan Saat, Stephan Aier, and Bettina Gleichauf. Assessing the Complexity of Dynamics in Enterprise Architecture Planning-Lessons from Chaos Theory. In Robert C. Nickerson and Ramesh Sharda, editors, *Proceedings of the 15th Americas Conference on Information Systems, AMCIS 2009, San Francisco, California, USA, August 6-9, 2009.* Association for Information Systems, 2009.

Pallab Saha, editor. *A systemic perspective to managing complexity with enterprise architecture.* Advances in business information systems and analytics (ABISA) book series. Business Science Reference, Hershey, PA, 2014. ISBN 9781466645196.

Vallabh Sambamurthy, Anandhi Bharadwaj, and Varun Grover. Shaping agility through digital options: Reconceptualizing the role of information technology in contemporary firms. *MIS Quarterly*, 27(2):237–263, 2003.

Ron Sanchez. Preparing for an uncertain future: Managing organizations for strategic flexibility. *International Studies of Management & Organization*, pages 71–94, 1997. ISSN 0020-8825.

Pier Paolo Saviotti and G. S. Mani. Competition, variety and technological evolution: a replicator dynamics model. *Journal of Evolutionary Economics*, 5(4):369–392, 1995. ISSN 0936-9937.

Roger C. Schank and Robert P. Abelson. *Scripts, plans, goals, and understanding: An inquiry into human knowledge structures*. Psychology Press, 2013. ISBN 1134919662.

Jaap Schekkerman. *Enterprise architecture good practices guide: how to manage the enterprise architecture practice*. Trafford, 2008. ISBN 1425156878.

Joachim Schelp and Robert Winter. Language communities in enterprise architecture research. In Vijay Vaishanvi and Sandeep Purao, editors, *Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology (DESRIST)*, 2009. ISBN 978-1-60558-408-9. doi: 10.1145/1555619.1555650.

Christian Schmidt. Business Architecture Quantified: How to Measure Business Complexity. In Daniel Simon and Christian Schmidt, editors, *Business Architecture Management*, Management for Professionals, pages 243–268. Springer International Publishing, 2015. ISBN 978-3-319-14570-9. doi: 10.1007/978-3-319-14571-6\textunderscore13.

Christian Schmidt and Peter Buxmann. Outcomes and success factors of enterprise IT architecture management: empirical insight from the international financial services industry. *European Journal of Information Systems*, 20(2):168–185, 2010. doi: 10.1057/ejis.2010.68.

Manfred Schmidt-Schauß and Gert Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991. ISSN 00043702.

Scott L. Schneberger and Ephraim R. McLean. The complexity cross: implications for practice. *Communications of the ACM*, 46(9):216–225, 2003. ISSN 00010782.

Alexander W. Schneider and Florian Matthes. Using Orientor Theory for Coherent Decision Making for Application Landscape Design. In Daniel Krob, Frédéric Boulanger, Gérard Morel, and Jean-Claude Roussel, editors, *Proceedings of the Poster Workshop at the 2014 Complex Systems Design and Management Conference*, 2014a.

Alexander W. Schneider and Florian Matthes. Unternehmensarchitekturgestütztes Controlling zur Beherrschung der IT-Komplexität. *Zeitschrift für Controlling*, 26(12):694–699, 2014b.

Alexander W. Schneider and Florian Matthes. Evolving the EAM Pattern Language. In *Proceedings of the 20th European Conference on Pattern Languages of Programs*. (to appear), 2015.

Alexander W. Schneider, Marin Zec, and Florian Matthes. Adopting Notions of Complexity for Enterprise Architecture Management. In *Proceedings of the 20th Americas Conference on Information Systems (AMCIS)*, 2014.

Alexander W. Schneider, Anna Gschwendtner, and Florian Matthes. Using System Dynamics Models to Understand and Improve Application Landscape Design. In O. Thomas and F. Teutenberg, editors, *Proceedings der 12. Internationalen Tagung Wirtschaftsinformatik (WI 2015)*, 2015a.

Alexander W. Schneider, Anna Gschwendtner, and Florian Matthes. *IT Architecture Standardization Survey*. Technische Universität München, Munich, Germany, 2015b.

Alexander W. Schneider, Thomas Reschenhofer, Alexander Schuetz, and Florian Matthes. Empirical Results for Application Landscape Complexity. In *48th Hawaii International Conference on System Sciences (HICSS)*, 2015c.

Alexander W. Schneider, Christian M. Schweda, and Florian Matthes. Identifikation überalterter Komponenten in einer IT-Architektur. In *Tagungsband der Multikonferenz Wirtschaftsinformatik (MKWI)*. under review, 2016.

Marguerite Schneider and Mark Somers. Organizations as Complex Adaptive Systems: Implications of Complexity Theory for Leadership Research. *The Leadership Quarterly*, 17(4): 351–365, 2006. ISSN 1048-9843.

Marten Schönherr. Towards a common terminology in the discipline of enterprise architecture. In Stephan Aier, Pontus Johnson, and Joachim Schelp, editors, *Pre-Proceedings of the 3rd Workshop on Trends in Enterprise Architecture Research*, pages 107–123, 2008.

Alexander Schuetz, Thomas Widjaja, and Jasmin Kaiser. Complexity in Enterprise Architectures - Conceptualization and Introduction of a Measure from a System Theoretic Perspective. In *21st European Conference on Information Systems (ECIS)*, 2013.

Maung Sein, Ola Henfridsson, Sandeep Purao, Matti Rossi, and Rikard Lindgren. Action design research. *MIS Quarterly*, 35(1):37–56, 2011.

Peter Senge. The fifth discipline: The art and science of the learning organization. *New York: Currency Doubleday*, 1990.

Claude Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27(3):379–423, 1948.

Carl Shapiro. The 2010 horizontal merger guidelines: From hedgehog to fox in forty years. *Antitrust Law Journal*, pages 49–107, 2010. ISSN 0003-6056.

William F. Sharpe. Microcomputer Perspectives: Economies of Scale, Technological Progress and Standardization. *Financial Analysts Journal*, pages 25–27, 1983. ISSN 0015-198X.

Walter Andrew Shewhart and William Edwards Deming. *Statistical method from the viewpoint of quality control*. Courier Corporation, 1939. ISBN 0486652327.

Kultar Singh. *Quantitative social research methods*. Sage Publications, Los Angeles, 2007. ISBN 8132101197.

Adam Smith and Andrew S. Skinner. *The wealth of nations*. World Scientific, 1991.

Floréal Solé, Richard Smith, Tiphaine Coillot, Eric de Bast, and Thierry Smith. Dental and tarsal anatomy of ‘ Miacis ’ latouri and a phylogenetic analysis of the earliest carnivoraforms (Mammalia, Carnivoramorpha). *Journal of Vertebrate Paleontology*, 34(1):1–21, 2014. ISSN 0272-4634. doi: 10.1080/02724634.2013.793195.

Dean R. Spitzer. *Transforming performance measurement: Rethinking the way we measure and drive organizational success*. American Management Association, New York, 2007. ISBN 0814430090.

Herbert Stachowiak. *Allgemeine Modelltheorie*. Springer-Verlag, Wien, New York, 1973. ISBN 9783211811061.

John Sterman. *Business dynamics: Systems thinking and modeling for a complex world*. Irwin/McGraw-Hill, Boston, 2000. ISBN 9780072387377.

John D. Sterman. Learning in and about complex systems. *System Dynamics Review*, 10(2-3): 291–330, 1994. ISSN 1099-1727. doi: 10.1002/sdr.4260100214.

John D. Sterman. All models are wrong: reflections on becoming a systems scientist. *System Dynamics Review*, 18(4):501–531, 2002. ISSN 1099-1727.

Stanley S. Stevens. On the Theory of Scales of Measurement. *Science*, 103(2684):677–680, 1946. ISSN 0036-8075. doi: 10.1126/science.103.2684.677.

Donald V. Steward. The design structure system: a method for managing the design of complex systems. *IEEE transactions on Engineering Management*, 28(3):71–74, 1981. ISSN 0018-9391.

Andrew Stirling. Diversity and ignorance in electricity supply investment: Addressing the solution rather than the problem. *Energy Policy*, 22(3):195–216, 1994. ISSN 0301-4215.

Andrew Stirling. On the economics and analysis of diversity. *Science Policy Research Unit (SPRU), Electronic Working Papers Series, Paper*, 28:1–156, 1998.

Andy Stirling. A general framework for analysing diversity in science, technology and society. *Journal of the Royal Society Interface*, 4(15):707–719, 2007. ISSN 1742-5689.

Andy Stirling. Multicriteria diversity analysis: a novel heuristic framework for appraising energy portfolios. *Energy Policy*, 38(4):1622–1634, 2010. ISSN 0301-4215.

Sagar Sunkle, Suman Roychoudhury, and Vinay Kulkarni. Using Intentional and System Dynamics Modeling to Address WHYs in Enterprise Architecture. *Proceedings of the International Conference on Software Engineering and Applications*, 2013.

James Surowiecki. *The wisdom of crowds*. Random House LLC, 2005. ISBN 0307275051.

John Sweller. Cognitive load theory, learning difficulty, and instructional design. *Learning and instruction*, 4(4):295–312, 1994. ISSN 0959-4752.

Paul P. Tallon. A process-oriented perspective on the alignment of information technology and business strategy. *Journal of Management Information Systems*, 24(3):227–268, 2007. ISSN 0742-1222.

Paul P. Tallon and Alain Pinsonneault. Competing perspectives on the link between strategic information technology alignment and organizational agility: insights from a mediation model. *MIS Quarterly*, 35(2):463–486, 2011.

Toomas Tamm, Peter B. Seddon, Graeme Shanks, and Peter Reynolds. How Does Enterprise Architecture Add Value to Organisations? *Communications of the Association for Information Systems*, 28:141–168, 2011. ISSN 1529-3181.

Hüseyin Tanriverdi. Information technology relatedness, knowledge management capability, and performance of multibusiness firms. *MIS Quarterly*, 29(2):311–334, 2005.

David J. Teece, Gary Pisano, and Amy Shuen. Dynamic capabilities and strategic management. *Strategic Management Journal*, 18(7):509–533, 1997. ISSN 1097-0266.

The Open Group. TOGAF Version 9.1, 2011. URL `http://pubs.opengroup.org/architecture/togaf9-doc/arch/`.

UK Department of Energy. Privatising Electricity: Command 322, HMSO, London, UK, 1988.

Rini van Solingen and Egon Berghout. *The goal/question/metric method: A practical guide for quality improvement of software development*. McGraw-Hill, London and Chicago, 1999. ISBN 9780077095536.

Robert van Wessel, Pieter Ribbers, and and De Vries, Henk. On the effects of IS company standards on business process performance. In *4th Conference on Standardization and Innovation in Information Technology*, pages 254–267, 2005.

N. Venkatraman. IT-enabled business transformation: from automation to business scope redefinition. *Sloan management review*, 35:73, 1994. ISSN 0019-848X.

Iris Vessey and Kerry Ward. The Dynamics of Sustainable IS Alignment: The Case for IS Adaptivity. *Journal of the Association for Information Systems*, 14(6), 2013. ISSN 1536-9323.

Iris Vessey, Venkataraman Ramesh, and Robert L. Glass. Research in information systems: An empirical study of diversity in the discipline and its journals. *Journal of Management Information Systems*, 19(2):129–174, 2002. ISSN 0742-1222.

Jan vom Brocke, Alexander Simons, Bjoern Niehaves, Kai Riemer, Ralf Plattfaut, and Anne Cleven. Reconstructing the Giant: On the Importance of Rigour in Documenting the Literature Search Process. In Sue Newell, Edgar Whitley, Nancy Pouloudi, Jonathan Wareham, and Lars Mathiassen, editors, *Proceedings of the 17th European Conference on Information Systems (ECIS)*, 2009.

Jos L.M. Vrancken. Layered models in IT standardization. In *International Conference on Systems, Man and Cybernetics*, pages 3862–3865, Taipei, Taiwan, 2006. IEEE. ISBN 1424400996.

Kenneth N. Waltz. *Theory of international politics*. Waveland Press, 2010. ISBN 1478610530.

D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393 (6684):440–442, 1998. ISSN 0028-0836. doi: 10.1038/30918.

Warren Weaver. Science and Complexity. *American Scientist*, 536(36):536–544, 1948.

Jane Webster and Richard T. Watson. Analyzing the Past to Prepare for the Future: Writing a Literature Review. *MIS Quarterly*, 26(2):13–23, 2002.

Alain Wegmann. The Systemic Enterprise Architecture Methodology (SEAM). Business and IT Alignment for Competitiveness, 2002.

Peter Weill, Mani Subramani, and Marianne Broadbent. Building IT infrastructure for strategic agility. *Sloan management review*, 44, 2002. ISSN 0019-848X.

Simon Weiss, Stephan Aier, and Robert Winter. Institutionalization and the Effectiveness of Enterprise Architecture Management. In Richard Baskerville and Michael Chau, editors, *Proceedings of the 34th International Conference on Information Systems*, 2013.

Martin L. Weitzman. On diversity. *The Quarterly Journal of Economics*, 107(2):363–405, 1992. ISSN 0033-5533.

Birger Wernerfelt. A resource-based view of the firm. *Strategic Management Journal*, 5(2): 171–180, 1984. ISSN 1097-0266.

Norbert Wiener. *Cybernetics or Control and Communication in the Animal and the Machine*. MIT press, 1965. ISBN 026273009X.

Thomas Wilde and Thomas Hess. Forschungsmethoden der Wirtschaftsinformatik. *Wirtschaftsinformatik*, 49(4):280–287, 2007. ISSN 0937-6429. doi: 10.1007/s11576-007-0064-z.

Robert Winter and Ronny Fischer. Essential Layers, Artifacts, and Dependencies of Enterprise Architecture. *Journal of Enterprise Architecture*, 2007.

L. A. Wojcik and K. C. Hoffman. Systems of Systems Engineering in the Enterprise Context: A Unifying Framework for Dynamics. In *International Conference on System of Systems Engineering*, pages 22–29, 2006. doi: 10.1109/SYSOSE.2006.1652268.

Stephen Wolfram. Universality and complexity in cellular automata. *Physica D: Nonlinear Phenomena*, 10(1):1–35, 1984. ISSN 0167-2789.

Stephen Wolfram. *Cellular automata and complexity: collected papers*. Addison-Wesley Reading, 1994.

Sewall Wright. Correlation and causation. *Journal of agricultural research*, 20(7):557–585, 1921.

Michael D. Wybo and Dale L. Goodhue. Using interdependence as a predictor of data standards. Theoretical and measurement issues. *Information & Management*, 29(6):317–329, 1995. ISSN 0378-7206.

Ling Ying-Hui, Wang Li-Sheng, Guo Xiao-Fei, Xiang Hao, Zhang Yun-Hai, Ding Jian-Ping, Zhang Zi-Jun, and Zhang Xiao-Rong. Assessment of genetic diversity among the twelve Chinese meat goat breeds using Weitzman approach. *JAPS, Journal of Animal and Plant Sciences*, 24(4):986–990, 2014. ISSN 1018-7081.

John A. Zachman. A framework for information systems architecture. *IBM systems journal*, 26 (3):276–292, 1987. ISSN 0018-8670.

Mohammad Esmaeil Zadeh, Gary Millar, and Edward Lewis. Mapping the Enterprise Architecture Principles in TOGAF to the Cybernetic Concepts–An Exploratory Study. In *45th Hawaii International Conference on System Science (HICSS)*, pages 4270–4276, 2012. doi: 10.1109/HICSS.2012.422.

Rüdiger Zarnekow, Walter Brenner, and Dipl-Ök Jochen Scheeg. Untersuchung der Lebenszykluskosten von IT-Anwendungen. *Wirtschaftsinformatik*, 46(3):181–187, 2004. ISSN 0937-6429.

Novica Zarvic and Roel J. Wieringa. An Integrated Enterprise Architecture Framework for Business-IT Alignment. In *Proceedings of the 18th International Conference on Advanced Information Systems Engineering (CAiSE'06)*, pages 262–270. Namur University Press,, 2006.

Abbreviations

ACID Atomicity, Consistency, Isolation, Durability

CLD Causal Loop Diagram

CEO Chief Executive Officer

COTS Commercial of the shelf

CSV Comma-separated values

DBMS Database Management System

DWH Data Warehouse

DL Description Logic

DSL Domain Specific Language

EA Enterprise Architecture

EAM Enterprise Architecture Management

FTEs Full Time Equivalents

GQM Goal-Question-Metric

IS Information System

ISR Information Systems Research

IT Information Technology

JEE Java Enterprise Edition

MxL Model-based Expression Language

OLAP On-line Analytical Processing

OLTP On-line Transactional Processing

OS Operating System

SD System Dynamics

SQL Structured Query Language

SFD Stock-and-Flow Diagram

TOGAF The Open Group Architecture Framework

UML Unified Modeling Language

US United States

VGM Viable Governance Model

VSM Viable System Model

XSLT Extensible Stylesheet Language Transformation

APPENDIX A

Appendix

Metric patterns observed in industry

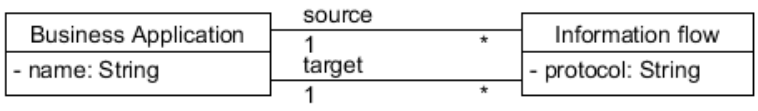
The following metric patterns have been observed in industry. Details about the observation process as well as additional information can be found in Chapter 5.1. The documentation follows the guidelines described by Matthes et al. (2012) and Schneider and Matthes (2015). If a certain metric is also described in literature, the respective source is provided in the intended section.

Number of interfaces per business application

Description

This measure indicates the number of in- and outgoing interfaces (information flows) of the business application to other business applications.

Information model



Organization-specific instantiation

Mapping:

Name in model	Mapped name	Contacts	Data sources
Business Application			
name			
source			
target			
Information flow			
protocol			

Properties:

KPI property	Property value	Observed values
Measurement frequency		
Interpretation		
KPI consumer		
KPI owner		
Target value		
Planned value(s)		
Tolerance value(s)		
Escalation rule		

Goals

Ensure compliance
Foster innovation
Improve capability provision
Improve project execution
Increase disaster tolerance
Increase homogeneity
Increase management satisfaction
Increase transparency
Reduce operating cost
Reduce security breaches

Calculation

Sum of numbers of interfaces that are in-, respectively outgoing from the business application to other business applications.

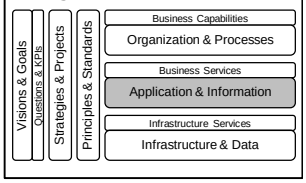
Code

EAM-KPI-0054

Sources

Mocker (2009)

Layers



Number of redundant business functions per business application

Description

This measure indicates the number of redundant business functions supported by a business application that are also covered by another business application.

Information model

Organization-specific instantiation

Mapping:

Name in model	Mapped name	Contacts	Data sources
Business Application			
name			
supports			
Business function			
name			

Properties:

KPI property	Property value	Observed values
Measurement frequency		
Interpretation		
KPI consumer		
KPI owner		
Target value		
Planned value(s)		
Tolerance value(s)		
Escalation rule		

Goals

Ensure compliance
Foster innovation
Improve capability provision
Improve project execution
Increase disaster tolerance
Increase homogeneity
Increase management satisfaction
Increase transparency
Reduce operating cost
Reduce security breaches

Calculation

Counting the number of business functions of a business application that are also supported by an another business application.

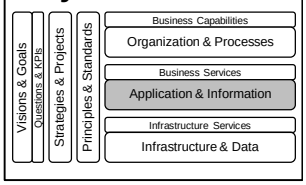
Code

EAM-KPI-0058

Sources

Mocker (2009)

Layers

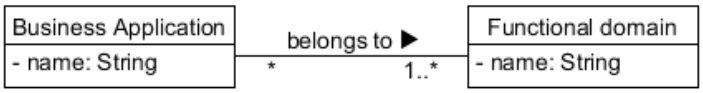


Number of used business applications by a functional domain

Description

This measure indicates the number of used business applications by a functional domain of a company.

Information model



Organization-specific instantiation

Mapping:

Name in model	Mapped name	Contacts	Data sources
Business Application			
name			
belongs to			
Functional domain			
name			

Properties:

KPI property	Property value	Observed values
Measurement frequency		
Interpretation		
KPI consumer		
KPI owner		
Target value		
Planned value(s)		
Tolerance value(s)		
Escalation rule		

Goals

Ensure compliance
Foster innovation
Improve capability provision
Improve project execution
Increase disaster tolerance
Increase homogeneity
Increase management satisfaction
Increase transparency
Reduce operating cost
Reduce security breaches

Calculation

Sum of number of used business applications by a functional domain of a company.

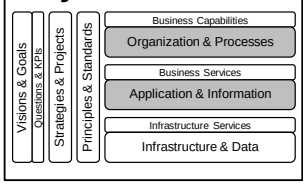
Code

EAM-KPI-0063

Sources

Schneider et al. (2015)

Layers

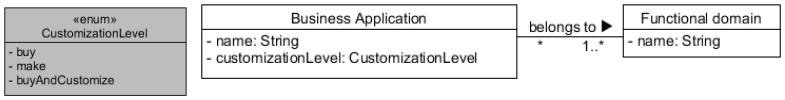


Standard conformity of business applications

Description

This measure indicates the standard conformity of business applications by classifying business applications of a domain according to their customization level. The customization level of a business application can be buy, make or buy and customize.

Information model



Organization-specific instantiation

Mapping:

Name in model	Mapped name	Contacts	Data sources
Business Application			
name			
customizationLevel			
belongs to			
Functional domain			
name			
CustomizationLevel			
buy			
make			
buyAndCustomize			

Properties:

KPI property	Property value	Observed values
Measurement frequency		
Interpretation		
KPI consumer		
KPI owner		
Target value		
Planned value(s)		
Tolerance value(s)		
Escalation rule		

Goals

Ensure compliance
Foster innovation
Improve capability provision
Improve project execution
Increase disaster tolerance
Increase homogeneity
Increase management satisfaction
Increase transparency
Reduce operating cost
Reduce security breaches

Calculation

For calculating the standard conformity for a specific domain, transform each business application's customization level to 1 (buy), 3 (make), or 5 (buyAndCustomize) and take the average according to their daily practice

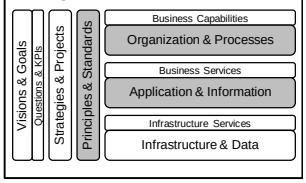
Code

EAM-KPI-0064

Sources

Schneider et al. (2015)

Layers

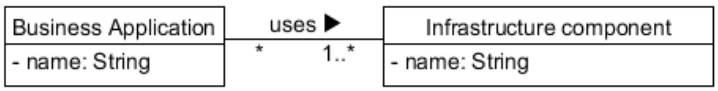


Number of infrastructure components used by business application

Description

This measure indicates the number of infrastructure components used by a specific business application.

Information model



Organization-specific instantiation

Mapping:

Name in model	Mapped name	Contacts	Data sources
Business Application			
name			
uses			
Infrastructure component			
name			

Properties:

KPI property	Property value	Observed values
Measurement frequency		
Interpretation		
KPI consumer		
KPI owner		
Target value		
Planned value(s)		
Tolerance value(s)		
Escalation rule		

Goals

Ensure compliance
Foster innovation
Improve capability provision
Improve project execution
Increase disaster tolerance
Increase homogeneity
Increase management satisfaction
Increase transparency
Reduce operating cost
Reduce security breaches

Calculation

Sum of numbers of infrastructure components which are used by a business application.

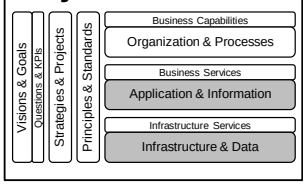
Code

EAM-KPI-0065

Sources

Schneider et al. (2015)

Layers

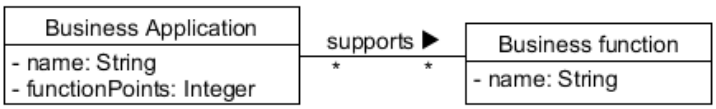


Functional scope of a business application

Description

This measure indicates the functional scope of a business application.

Information model



Organization-specific instantiation

Name in model	Mapped name	Contacts	Data sources
Business Application			
name			
functionPoints			
supports			
Business function			
name			

Properties:

KPI property	Property value	Observed values
Measurement frequency		
Interpretation		
KPI consumer		
KPI owner		
Target value		
Planned value(s)		
Tolerance value(s)		
Escalation rule		

Goals

Ensure compliance
Foster innovation
Improve capability provision
Improve project execution
Increase disaster tolerance
Increase homogeneity
Increase management satisfaction
Increase transparency
Reduce operating cost
Reduce security breaches

Calculation

For calculating the functional scope of a business application, one can take the functionPoints of a business application or the sum of business functions which are supported by the business application.

Code

EAM-KPI-0066

Sources

Schneider et al. (2015)

Layers