CONCEPTUAL INDEPENDENCE AS AN ENABLER OF IT FLEXIBILITY

Debbie Tarenskeen

Conceptual independence as an enabler of IT Flexibility

Deborah Tarenskeen

ISBN/EAN 978-94-6423-099-4
NUR-code 982

Cover design by Hans Davids
Cover photograph based on photo of Princess Máxima Center, Utrecht
Lay-out by Hans Davids
Published by ProefschriftMaken, www.proefschriftmaken.nl, 2021

Conceptual independence as an enabler of IT Flexibility

De invloed van conceptual independence op de flexibiliteit van IT infrastructuren
(met een samenvatting in het Nederlands)

Proefschrift

ter verkrijging van de graad van doctor aan de
Universiteit Utrecht
op gezag van de
rector magnificus, prof.dr. H.R.B.M. Kummeling,
ingevolge het besluit van het college voor promoties
in het openbaar te verdedigen op

maandag 1 februari 2021 des middags te 2.30 uur

door

Deborah Tarenskeen

geboren op 10 maart 1953
te Den Haag

Promotoren:

Prof. dr. S. Brinkkemper

Prof. dr. ir. S.M.M. Joosten

Copromotoren:

Dr. R. van de Wetering

Dr. ir. R. Bakker

Dit proefschrift werd (mede) mogelijk gemaakt met financiële steun van
HAN University of Applied Sciences, Arnhem.

Preface

This work could not have been realized without extensive support and effort of professionals and friends that believed in the project during the many years since 2012.

My plan to study the IT systems in organizations in healthcare has been made possible by IT architects that opened doors for me to the Princess Máxima Center, Ziekenhuis Rivierenland, the Carintreggeland organization and GGZ Nederland. I am grateful to Ronald de Vries, René Verschoor, Jos Geesken, Richard Derks, Cees van der Zwaag and Martin van der Meer of Code24, not only for their inspiring words, but also for their support in organizing meetings with IT experts.

During the project I have been lucky to meet with many researchers at the Landelijk architectuur congres (LAC) and the openGroup conferences for sharing their thoughts and doubts with me. Many conversations have not been included in the research, e.g. the contacts with Midwifery Netherlands (KNOV) and Angela Verbeeten.

Theoretical insights were formed in discussions with Simon McGinnes in Exeter about conceptual independence and in meetings with Nam Suh and scientists at the conferences of axiomatic design.

I thank friends, colleagues and students that have contributed, Christian Köppe, for introducing me to Viking PLOP and Willy Koning, Emiliano Heyns, Laura Kaiser, Albert Geurtsen and Ria van Kessel for helping out with chapter 7. They have read and evaluated lists of terms that pop up in open source software to determine whether they were healthcare-related or not. Students of the Bachelor of ICT study developed prototypes and tools for application in the research. I would also like to thank Ulrike Goossen for accompanying me to European conferences and being a good friend.

Then, in the last stage of the project, many professionals in IT in mental healthcare (GGZ) gave their time and informed me about their work. The interviews with these IT professionals provided essential information for the study.

I really appreciate the time and support of Hans Davids, who designed the book cover and set the text in the new lay-out, and the advice of my two aces in English language, Lidy and Niko Berndsen, who helped to improve the text of the publication.

From the start and during the course of the project informal discussions with Cor Baars were very valuable for me. And last but not least, I am grateful to the supervisors that have guided me into the scientific community, René Bakker, Rogier van de Wetering, Sjaak Brinkkemper, Stef Joosten and Stijn Hoppenbrouwers. These supervisors were tireless in reading my papers in different versions and providing helpful feedback.

Thank you all.

Contents

1 Introduction

1.1. Motivation

At the beginning of the millennium it became clear that the healthcare sector was confronted with a changing environment. The changes varied from changes in demographics (Schäfer et al., 2010; Van den Berg Jeths, Timmermans, Hoeymans, & Woittiez, 2004), new visions on healthcare, changes in medical knowledge and a growing need for innovative IT to support the healthcare processes. Pressures on the healthcare sector existed in 2010 due to reforms in financial aspects and changing roles of patients, as described by Schäfer et al. (Schäfer, et al., 2010). Priority was given by IT architects in healthcare to the alignment of the care and cure processes and enterprise architecture (EA) (Henderson & Venkatraman, 1993).

Tamm et al. (Tamm, Seddon, Shanks, & Reynolds, 2011) describe EA as the definition and representation of a high-level view of an enterprise's business processes and IT systems, their interrelationships and the extent to which different parts of the enterprise share these processes and systems.

In research on EA from 2005 to 2010, EA appeared promising (Chan, Sabherwal, & Thatcher, 2006; Lange, Mendling, & Recker, 2012; Tamm, et al., 2011). Benefits were expected, according to Van der Raadt et al.: "Enterprise Architecture (EA) is increasingly being used by large organizations to get a grip on the complexity of their business processes, IS and technical infrastructure." (Van der Raadt, Bonnet, Schouten, & Van Vliet, 2010). An empirical, critical study from 1994 indicated that application in practice was difficult (Y.-G. Kim & Everest, 1994).

The ideas behind EA were inspired by Henderson and Venkatraman (Henderson & Venkatraman, 1993), who analyzed the relationship between business strategy and IT, thereby moving IT to a strategic level. They observed that companies failed to extract value from IT and suggested the cause might be a lack of alignment between business and IT strategies, B/IT alignment. The B/IT alignment therefore became the starting point of our research.

A need for EA methods and conceptual models for alignment of B/IT in organizations emerged. The OpenGroup (TheOpenGroup, 2011) published an extensive method, the Architecture Development Method (ADM), TOGAF, for application in organizations. Starting from the requirements analysis, different architectures were described for which alignment of the business architecture, IS architecture and technology architecture was crucial. A solution architecture that supported business processes needed to be developed. The solution architecture is seen by the OpenGroup (TheOpenGroup, 2011) as: "A description of a discrete and focused business operation or activity and how IS/IT supports that operation. The architecture typically applies to a single project or project release, assisting in the translation of requirements into a solution vision, high-level business and/or IT system specifications and a portfolio of implementation tasks."

In healthcare an extensive case study for applying TOGAF in a healthcare organization, Chiu at al. (Chiu, Kohli, & Chaczko, 2012) reported on positive outcomes for applying TOGAF, concluding that: "The Smart Hospital System meets the health industry's need for consolidated patient records through the use of TOGAF by reducing the need for rework on legacy applications to fit into the new unified framework." A perspective unfolds that the new EA aids in adapting the IT systems to scalability and extensibility. However, "this is achieved by investigating the candidate architecture models, frameworks and patterns that match the category of integration facilitators, to ensure that the health institution's demands for scalability and extensibility are realized." as Chiu et al. (Chiu, et al., 2012) describe. A substantial effort is needed to comply with the guidelines of TOGAF for the complete IT infrastructure. The IT infrastructure is viewed as the whole of organizational IT components and the relations and dependencies between the components.

Other difficulties for applying TOGAF were described. Research literature Franke et al. (Franke, Ekstedt, Lagerström, Saat, & Winter, 2010) and Aier (Aier, 2014) explained that it was generally assumed that TOGAF required a steep learning curve, consistent use of terminology and was difficult. Hence, TOGAF presumably, was not suited for small organizations.

We notice high expectations of EA in the Netherlands, especially in healthcare, as published in 2010 by a branch organization of IT professionals detailing guidelines for IT architecture in healthcare organizations (Gondelach et al., 2010). Summarizing the early literature on EA and B/IT alignment we find that the flexibility of IT played a subordinate role. The attention for IT flexibility increased in later research, see section 1.3 subsection IT flexibility and IT infrastructure.

1.2. Main focus: Flexibility of IT infrastructure in healthcare organizations

Our study explores the approaches of the enterprise architects and IT architects in healthcare. We have expected to find adapted, simplified TOGAF development methods for direct application in practice. In real-life cases we observed that hospitals and other healthcare organizations did not start the IT activities in a green field. IT infrastructures already existed in the organization and were in operation. The existing software was tried and tested for supporting the current care processes and did not anticipate future changes. Enterprise architects were expected to collect new requirements and to develop an existing and a desired business architecture for the organization. These architectures were a necessary precondition for building the solution architecture according to the guidelines of TOGAF.

The IT infrastructure can be seen as the installed base, grown to fit the current requirements and demands. The installed base is seen as an integrated whole of IT systems in which work practices and organizational context are embedded by Aanestad et al. (Aanestad, Grisot, Hanseth, & Vassilakopoulou, 2017a). In another study Aanestad et al. (Aanestad, Grisot, Hanseth, & Vassilakopoulou, 2017b) assess that European countries have reached such a level of technological maturity that healthcare processes are supported in a satisfactory way. However, new demands such as communication between systems are not anticipated in the systems. They prefer the term information infrastructure instead of IT or IS infrastructure, because they focus on communication instead of IT infrastructure. Changing the information infrastructure will stir up reactions of supporters and opponents of change. In practice, we found in the cases that the installed base does contain systems in which work practices and organizational context are embedded. The installed bases in the case studies did exhibit the characteristics that Aanestad et al. describe.

Following Aanestad et al. we concluded that the design was not ready for new requirements. The focus of our research shifted from EA to characteristics of existing systems capable of change. This study describes the theory and multiple cases about realizing changes in IS that rigidly support healthcare processes that are no longer adequate.

Observations of the approaches and failing attempts of enterprise architects and IT architects to map requirements on components of an existing IT infrastructure, created doubt that IT flexibility could be achieved with methods similar to the TOGAF approach. In research, however, it has been described that IT flexibility could be influenced in real-life organizations and that it contributed to firm performance and competitive market advantage (Mikalef, Pateli, & Van de Wetering, 2016; Van de Wetering, Mikalef, & Pateli, 2018). Because of numerous indications in research literature that the influence of software structure on IT flexibility could be substantial, see section 1.4, Reusability and software systems evolution, this research focuses directly on the influence of underlying software structures and other IT resources for enhancing IT flexibility. IT resources in the IT infrastructure consist of IT hardware, software and IT personnel in organizations. Broadbent et al. see IT infrastructure as a capability: “The major components of IT infrastructure are hardware platforms, base software platforms, communications technology, middleware and other capabilities that provide shared services to a range of applications and common handling mechanisms for different data types.” (Broadbent, Weill, & Neo, 1999). Kim et al. see IT capabilities as the capabilities of the organization to apply and coordinate the use of IT resources to turn them into business value (G. Kim, Shin, Kim, & Lee, 2011).

The main question in this study examines the influence of IT resources and other IT capabilities on the IT capability of IT flexibility. This study examines characteristics of IT resources such as IS software and its influence on IT flexibility and therefore mentions IT resources explicitly in the main research question.

1.3. Research domain

In this section enterprise architecture (EA), the relation between EA and axiomatic design (AD) and the relation between IT flexibility and conceptual independence (CI) will be described. The research is positioned in the intersection of these different research areas and bridges business/IT alignment by discussing the application of conceptual models in software.

Enterprise architecture. The research domain is positioned in EA. In the first stage of the study objectives for and benefits of applying EA in healthcare organizations were explored. As seen by Tamm et al. (Tamm, et al., 2011), direct and indirect benefits from EA are integrated in a proposed EA Benefits Model, EABM, based on literature study. Tamm et al. (Tamm, et al., 2011) expect that: "…large organizations with a complex IT environment, whose business model favors high levels of organization-wide standardization and integration, can expect to benefit the most from EA."

The tradition of EA is based on completeness, as modeled by Zachman, who describes a framework for all elements when conceptualizing IS architecture (Zachman, 1987). Completeness remained a central focus in EA, although problems have been signaled as early as 1994 in practice. It was found that detailing the IS architecture proved too time consuming (Y.-G. Kim & Everest, 1994). The field has been affected by a need for green field situations, in which the IS architecture could be built top down. Much later Kotusev (Kotusev, 2019) heavily criticized the application of the Zachman framework as a tool and questioned the fundamental assumptions of IS architecture as a tool for practice.

TOGAF as an architecture development method also builds on the Zachman tradition of defining and describing all requirements and intermediate architectures, the business architecture, the information architecture and the technological architecture (TheOpenGroup, 2011). TOGAF guides enterprise architects in the process of integrating different views in EA artifacts, thereby suggesting that the process of EA would improve communication and lead to a more aligned B/IT in the organization. However, managing only high-level requirements and business processes in EA in the case studies did not lead to satisfying results. It appeared necessary to also map lower-level functional requirements to low-level IT components in the IT infrastructure as in axiomatic design (AD), see next section.

Axiomatic design and EA. Axiomatic design (AD) explains and applies the relation between functional requirements and system components to construct systems that are well designed. AD describes principles for mapping functional requirements to system components. Because we extensively describe the ideas of axiomatic design (AD) in chapters 3 and 6, the author will only summarize the ideas of Suh here (Suh, 2001). The ideas of AD have been developed in the research domain of industrial design of products and product lines. Flexibility of products and product lines appeared greatly dependent on the way the products had been constructed. Products and product lines consisting of independent parts that could be replaced or adapted and integrated were seen as more adaptable for future purposes. The ideas of Suh resemble ideas in software development for flexible systems characterized by a modular structure of independent components.

Suh describes two principles for adaptable systems. The first principle, the independence axiom (in words of Suh (Suh, 2001)), can be described as follows: different independent components must be linked to functional requirements on each level of formulation.

Although the guidelines for adaptable systems according to Suh are transferrable to software systems as stated in a paper of Do and Suh (Do & Suh, 1999), components in software require a very detailed description of all functional requirements and depend on business information.

The second principle, the information axiom, states that all components must be designed and constructed by using as little information for the components as possible. Minimalizing the information for building each component will in theory lead to systems that are adaptable. Characteristics of IS related to business information, however, add complexity that is hard to reconcile with AD.

IT flexibility and IT infrastructure. Existing research shows how IT flexibility is a key factor in establishing the stable processes and capabilities needed for surviving in highly unstable conditions and still showing exceptional organizational performance. Van de Wetering et al. (Van de Wetering, et al., 2018) demonstrate how IT flexibility plays a prominent role in these capabilities of firms to meet the challenges the environment poses. IT flexibility originates from the theories of modular systems. Modular systems are complex systems consisting of modular or else nearly decomposable subunits that tend to evolve faster, increase the rate of adaptive response and tune towards stable, self-generating configurations, as formulated by Van de Wetering et al. (Van de Wetering, et al., 2018), following Simon (Simon, 1962). The definition of IT flexibility that is applied in this study has evolved from a long tradition of research in this field, cf. Duncan, Byrd et al., Simon, Teece et al. (Byrd & Turner, 2000; Duncan, 1995; Simon, 1962; Teece, Pisano, & Shuen, 1997). See for a detailed explanation chapters 8 and 9.

We study IT flexibility, here, in organization-wide IT infrastructures that include next to the IS, all resources as networks of interconnected hardware and software, security and cloud functionality for supporting the organization. Definitions of IT infrastructure lead back to the Resource Based View (RBV), where empirical research has been performed to study how combinations of resources allow firms in competitive markets to maintain a competitive advantage (Barney, 2001; Broadbent, et al., 1999; Ray, Barney, & Muhanna, 2004).

IT resources can be applied in a variety of ways as is argued by Eisenhardt and Martin (Eisenhardt & Martin, 2000), Teece et al. (Teece, et al., 1997), Pavlou and El Sawy (Pavlou & El Sawy, 2011). The application and use of IT resources will lead to different outcomes. In volatile markets firms that develop dynamic capabilities (DC) make full use of unique combinations of resources for staying alive in environments that force them to change.

Dimensions of IT infrastructure flexibility. Dimensions of IT infrastructure can be examined empirically in organizations as is shown by Van de Wetering et al. (Van de Wetering, et al., 2018), Mikalef et al. (Mikalef, et al., 2016) and Van de Wetering (Van de Wetering, 2019). They conceptualize and operationalize IT flexibility with the following dimensions: modularity, transparency, standardization and scalability. In Table 1.1. we present the definitions we will apply in this research.

Table 1.1 IT Flexibility according to Mikalef, Patelli and Van de Wetering.

IT Flexibility Dimensions	Description
Modularity	"Loose coupling" is the main idea behind modularity
Transparency	Making IT systems behave as one integrated system with seamless accessibility to functions and data
Standardization	Applying organization wide standards for hardware and software
Scalability	Measures how well the IT Infrastructure can be scaled up and upgraded, when adaptation is necessary due to growing demand and increasing number of users

Source: IT Flexibility according to Mikalef, Patelli and Van de Wetering (Mikalef, et al., 2016).

Conceptual independence and IT flexibility. The dimensions of IT flexibility in Table 1.1. do not refer to specific characteristics of IS, systems that play a significant role in supporting healthcare and organizational processes by delivering information to support the professionals. In healthcare, how to handle meaningful information has been a subject of discussions from the start of digitalization of data. The effects of certain implementations of meaningful concepts in IT applications on IT flexibility has not been researched yet. McGinnes and Kapros claim that conceptual independence (CI) as a design principle for IS improves adaptability of IS, see section 1.4. By examining CI this study offers further insights in the impact of implementations of meaningful concepts on IT flexibility.

IT outsourcing. The role of the supplier partner in IT outsourcing has often been neglected in studies on IT flexibility. Case studies and surveys research large organizations that often deploy their own custom-made software. In healthcare in the Netherlands, however, even large healthcare organizations purchase commercial-off-the-shelf software (COTS), standard software customized for the specific organizations. We assumed the IT outsourcing partner would exercise influence on IT flexibility. Therefore, we have examined if and how CI in combination with IT outsourcing strategies influences IT flexibility (Bui, Leo, & Adelakun, 2019; Dibbern, Goles, Hirschheim, & Jayatilaka, 2004).

1.4. Conceptual independence and reusability of software

In this section the principle of CI will be described in the context of a long line of research of reusability of software. Traditionally the flexibility of IS has been researched in database systems and systems that depend on feedback from business actors in the organization. In this section we create a new perspective by focusing on the implementation of conceptual models in IS as a factor for reusability of software.

Reusability and software systems evolution. A long history exists of studies on difficulties in changing software code, when viewed from a business perspective (Garlan, Allen, & Ockerbloom, 1995, 2009; Lehman, 1996). Literature about reuse and evolvability of software contained many chapters about inherent inflexibility in software. There are implemented detailed connections in software that obstructed reuse (Garlan, et al., 1995, 2009).

Older research by Lehman (Lehman, 1996) hypothesized about general laws in software that obstruct flexibility when the software systems evolve. The laws in software evolution explain how specific software systems, E-type systems, which fulfill a role in real-life situations, must be continuously adapted. The systems tend to increase in complexity and decrease in flexibility in the course of time. Software of E-type systems is characterized as needing feedback processes for construction and maintenance. E-type systems that are used in real-life situations have to adapt to changes in technical, social and managerial practices to be capable of supporting the organizations. Lehman emphasizes the feedback system in global industrial software processes. “It involves not only technical development but process engineering, many levels of management, corporate executes, marketing, user support and so on.” He states that the output of the feedback system is a complex function of processes in the organization that involve technological, information and social factors and a diversity of agencies.

His study of software systems has recently been restudied by Skoulis et al. (Skoulis, Vassiliadis, & Zarras, 2014). They observed that Lehman’s analysis of software systems held for a long time, for his ideas “were reviewed and enhanced for nearly 40 years”. Their findings demonstrate that schema evolution relies on a feedback-regulated system, because it shows both the need for “expanding its information capacity to address user needs and the need to control the unordered expansion, with perfective maintenance.” Although continuous growth and continuous increase in complexity were not confirmed, changes in spikes were observed.

Other studies in data intensive software systems led to similar conclusions. New adaptations were difficult to implement as seen by Qiu et al. (Qiu, Li, & Su, 2013) because of many interdependencies between schemas and code. Goeminne found difficulties by studying the effect of changes in database systems and frameworks (Goeminne, Decan, & Mens, 2014). Hidden dependencies caused by data base schemas were researched by Ajienka et al. (Ajienka, Capiluppi, & Counsell, 2017). Illustrations of labor intensive work for change request can be found in Kagdi and Poshyvanyk (Kagdi & Poshyvanyk, 2009). Database schema evolution is described by Vassiliadis et al. (Vassiliadis, Zarras, & Skoulis, 2015). They find that early changes appear more frequently than later changes. In a newer study Vassiliadis et al. believe that the observed evidence strongly indicates “that databases are rigidity-prone rather than evolution-prone.” (Vassiliadis, Zarras, & Skoulis, 2017).

Zickert et al. (Zickert & Beck, 2012) studied adaptability of existing IS and concluded that “existing systems increase complexity due to conflicting interdependencies”. Moreover, top down approaches only increased complexity. These studies are particularly relevant for EA practices that advise top down change processes.

Conceptual independence. Lehman and others after him concluded that meaningful terminology is embedded in the database schemas of data intensive systems referring to context terms in the situation where software systems were applied. McGinnes and Kapros (McGinnes & Kapros, 2015) developed design principles for IS that counterbalanced changes in such a way that low-level editing of application code would not be necessary. The design principles are directed towards the implementation of conceptual models in software. The design principle of Conceptual Independence (CI) states that adaptable software is structured in such a way that the conceptual models have been implemented separately from the application logic (McGinnes & Kapros, 2015), which means that conceptual models can be changed in software without having to reprogram the application code. It also follows that if data is ordered and labeled to the relevant conceptual model, different (versions of) conceptual models can be deployed next to each other with the same software application.

The role of conceptual models in software development is of the utmost importance. The structure of domain concepts and terminology is vital for users, for they must be able to recognize communication in the IS whether coming from colleagues or other professionals. Conceptual models in software development are the basis for the construction of data models. The research of Lehman and others after him focused on database schemas, cf. the previous section. These are the representations of conceptual models in data-intensive systems. Data-intensive software applications need data models for ordering and processing data in systems and managing program flow. Therefore the development of conceptual models is a necessary step in software development. Conceptual models are also directly linked to users’ understanding of the system and the functional requirements. However, the literature on the flexibility of enterprise-wide IT architectures prioritizes the technical quality attributes of IT infrastructures (Bachmann et al., 2011; Bass, Clements, & Kazman, 2012) and does not mention flexibility of conceptual models.

The design principle of CI has been detailed in 6 principles for software applications (McGinnes & Kapros, 2015). We base the theoretical and practical research on the theoretical work of McGinnes regarding CI (McGinnes, 2011; McGinnes & Kapros, 2015). Our aim is to research its relevance for IT infrastructure flexibility.

The openEHR specification. For studying the implementations of CI we have resorted to the openEHR standard. The ideas for openEHR have resulted from work executed in the GEHR project (Cordis EU, 2014). The Good Electronic Health Record project of the European Commission was started in 1992 and ended in 1994. It aimed to provide the framework for the storage of and access to all of the clinical information required for the provision of patient health care with a multi-lingual dictionary of health record items. Although the project resulted in useful descriptions of architecture, the end documents were not ready for application in practice. The ISO standard 18308, Health informatics - Requirements for an electronic health record architecture, was developed based on results from GEHR (Beale, 2002, 2003). The openEHR community continued the work and brought the specification to the public domain. The specification, as a meta-standard, is meant to support modeling of medical knowledge for application in software to support medical professionals (openEHR Foundation, 2020b). For flexible modeling of medical knowledge the openEHR standard applies archetypes in a format suitable for the openEHR software. Beale (Beale, 2002; Beale & Heard, 2007) developed the standards for archetypes that are representations of medical concepts. With archetypes it became possible to use different archetypes in one IS or to deploy different versions of archetypes next to each other. An international community supports the standard and develops it further. The international support is growing (openEHR Foundation, 2020a, 2020b).

We argue in chapters 8 and 9 that openEHR can be viewed as a representative of CI. We study implementations of openEHR in the cases as a proxy for implementations of CI.

1.5. Research questions

Main research question. By studying the work and efforts of enterprise architects with the task of mapping functional requirements to existing system components, it became clear that information about the functionality of software components was lacking. The lack of information impeded the EA process. IT architects found that for adapting IS more information about the content in the IS was necessary.

After the first observational stage the choice was made to elaborate further on the concept of IT flexibility. The term IT flexibility had not been strictly defined in this research during the first case studies, because the objective has been to explore the way IT architects manage changes in practice. A working definition of IT flexibility can be described as “the capability of IT architects to change existing IT infrastructure systems in case of changing demands from the healthcare organization or linked to external influences such as social and technical developments.” Since the IT architects and healthcare organizations deal with a broad spectrum of requests, they have the responsibility to prioritize requirements and eventually have to evaluate the degree of IT flexibility needed.

Later, definitions of IT flexibility as found in theory were applied focusing on enablers and factors that could improve IT flexibility while pursuing the main research question of this dissertation:

“Which IT resources and other IT capabilities contribute to IT flexibility in healthcare organizations in a changing environment in the Netherlands?”

The main research question in this study has been divided into four subthemes related to the research questions RQ1, RQ2, RQ3 and RQ4.

Theme 1: Models to support enterprise architects. In professional practice journals, IT architects express difficulties with adapting systems to new business and functional requirements. The first research question observed and explored the EA activities that enterprise architects performed. Hence, we define RQ1: *"Which models can support IT architects in practice with their work of developing new architectures for existing IT infrastructures?"*

The research had been started by developing different versions of a model that could support enterprise architects with EA based on their own expressed information needs. The model is described in chapter 2. Then the different views of EA were explored. Business architects and technical IT architects in hospitals and a care institute explained or made diagrams of the structure of their IT architecture. The idea in this study that functional requirements can be mapped to independent components in the existing IT infrastructure has been described in theory in axiomatic design (AD). To explore if enterprise architecture models could be extended with AD we performed case studies in which enterprise architects and IT architects experimented with modeling IT infrastructure with AD as described in chapter 3.

Theme 2: State of affairs regarding IT flexibility in healthcare. The researchers questioned the flexibility of existing IT architectures of the IT systems in use. Hence we explore existing IT architecture as found in case studies of large healthcare organizations. We define RQ2: *"What is the state of affairs concerning the flexibility of IT architectures in large IT infrastructures in healthcare organizations in the Netherlands?"*

First, a follow-up study has been performed on the model of the enterprise architects in chapter 2, in a hospital in the process of formation. The reactions of the stakeholders in the organization have been discussed with enterprise architects and have been studied in documentation. The follow-up projects in the organization have been examined in chapter 4. In chapter 6 three medium to large healthcare organizations have reported on their ideas about modularity based on AD by exploring the demarcation lines in the IT infrastructures.

Theme 3: Design principles in software that enable flexibility in IS. Theme 3 reports on desk research for RQ3 regarding software designs that enable the flexibility of IS software: *"Which design principles in software engineering research literature enable flexibility in Information Systems?"*

Since, in practice, information about the software structure had not been available in the case organizations, we shifted attention to theoretical research to answer RQ3. This theoretical research on software adaptability led to a design of a combined model for AD and CI described in chapters 5 and 6.

Theme 4: Contribution of conceptual independence and outsourcing to IT flexibility. The last question, RQ4, returns to the study of IT flexibility in practice: *"Do the implementation of Conceptual Independence and IT outsourcing affect the IT infrastructure flexibility of the organization in healthcare?"* First, the presence of the principle of CI has been tested in open source software for EHR systems. The results would give an indication about the acceptance of CI in the open source software community, see chapter 7.

Then the study focused on case studies in healthcare organizations in the Netherlands. The literature on CI claims that CI improves flexibility of single IS. In the last chapters we study a stronger effect. The effects of CI on the organization-wide IT infrastructure flexibility have been researched in chapter 8. Other IT resources and capabilities affected by CI were considered. Since the IT outsourcing of the IS development function is common in healthcare in the Netherlands, IT outsourcing as an IT capability has been studied in chapter 9.

1.6. Research method

To study the main research question we have mainly applied case studies for exploration and testing of propositions. Case studies are relevant for studying IT flexibility in real-life situations. Therefore, we examined exemplars of hospital IT infrastructure, home care IT infrastructure and IT infrastructures of mental healthcare organizations. The case studies have been embedded in a larger research design that worked on developing artefacts and testing design principles in practice and building theory for the EA research domain and the IT flexibility research domain.

Theory building. Gregor sees theories in a broad sense (Gregor, 2006); she extends the narrow meaning of theory as a search for general laws, principles or causes to a meaning that “encompasses what might be termed elsewhere conjectures, models, frameworks or body of knowledge.” Knowledge that is developed in scientific enterprises “consists of those constructions about which there is a relative consensus (or at least some movement towards consensus) among those competent.” Explanations can include notions of causality “that do not depend on law-like generalizations or statistical association alone,” and can be embedded in a communicative process.

Gregor distinguishes five goals for scientific research: analysis and description, explanation, prediction and prescription. The first goal is concerned with the exploration of the territory of the phenomenon in the research domain and describes found related constructs. When possible explanations have been detected, the theory building proceeds with detailing the explanations and testing the soundness of the explaining narratives. In a further step the explanations have proved their worth and predictions can be made on the constructs involved in the research model. In the last goal the theory is able to prescribe certain behaviors for relevant actors to realize desired effects. In our study we have hypothesized that CI would affect IT infrastructure flexibility and tested this hypothesis. In future studies instruments could be developed to measure the use of specific meta-models or the presence of IT capabilities. With the instruments a prediction can be made regarding the perceived flexibility of the IT infrastructure.

In table 1.2. an overview is given of the performed research described in the different chapters in relation to categories of theory building of Gregor (Gregor, 2006).

The case studies that we have conducted with a variety of research methods were initially performed to extract EA models. The case studies in chapters 2 and 3 explored the working methods of enterprise architects to answer RQ1 and RQ2.

After the case study research, a desk research study has been performed for RQ3 to discover possible explanations for difficulties with IT flexibility. This research repositions the results from the domain of reuse of software to the domain of IT flexibility in organizations. With RQ4 the design principle, CI, for realizing adaptable IS has been tested in practice with case studies and a mix-methods approach. Propositions about the relation of CI to IT flexibility were formulated and tested, first in open source software, later in ten case studies. In chapter 9 the relation of IT outsourcing to IT flexibility has been explored.

Table 1.2. Overview Research Questions, Chapters and Theory Building

RQ	**Chapter**	**Contribution to theory building**		
		Analysis and description	Explanation	Explanation and Prediction
1	Chapter 2	●		
1	Chapter 3	●		
2	Chapter 4	●		
3	Chapter 5		●	
2,3	Chapter 6		●	
4	Chapter 7		●	
4	Chapter 8			●
4	Chapter 9	●		

Design science research. The first three cases were structured according to design science research (A. Hevner & Chatterjee, 2010; A. R. Hevner, 2007; A. R. Hevner, March, Park, & Ram, 2004; Wieringa, 2014). In Fig. 1.1 the framework of design science research of Hevner et al. (A. R. Hevner, et al., 2004) is represented. The framework positions design activities in the middle column. Design science research aims at studying questions that have a meaning for practice and are socially relevant, see first column Environment. Design science applies design and development activities for data collection and analyzes the resulting artefacts to find answers to research questions. Design activities often involve participants from practice. Findings from artefacts are compared to results of theoretical or empirical research and can refute, conform or complement research literature, thereby adding to the knowledge of the scientific domain, the Knowledge Base.

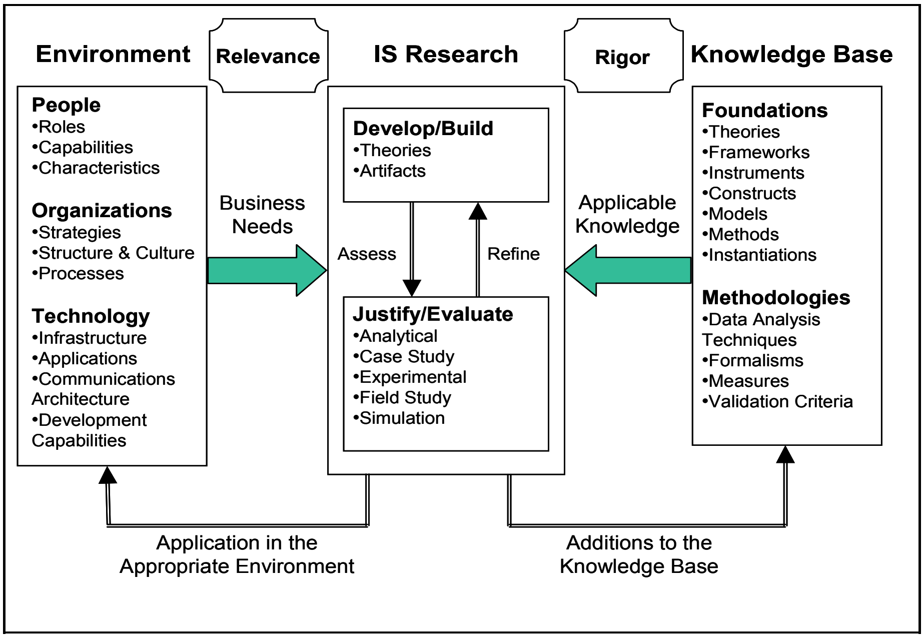


Fig. 1.1. Hevner's Research Framework

We have performed design activities in collaboration with the architects in the first three cases of the research where enterprise architects were observed and models of their information needs were developed. These activities can be positioned in the Develop/Build part. A first evaluation, Justify/Evaluate, in practice has been executed by evaluation of the enterprise architects of the models, a reflection on the EA model artifact and an analysis of its application in the organization. The results were not unambiguous and needed clarification. Thus we shifted attention to literature research in the knowledge base to find new entries to resume research in practice. The design science framework has been leading in our research for modeling EA in chapters 2, 3, 4 and modeling a design for a flexible IS in chapters 5 and 6. We have started with a relevant research question that arose from EA practice how to support enterprise and IT architects in healthcare with conceptual models and tooling in their work of mapping new requirements to an existing IT infrastructure. We have observed the architects in their work and proposed models that were iteratively adjusted.

After the artefacts did not prove to be suited for implementation in the organizations, the author decided to perform a desk research study for clarification of the bottlenecks. Findings from literature research were a lever for new follow-up research in practice on the relation between IT flexibility and underlying software structure in chapter 7.

In the next stage based on the combined findings from practice and literature we have formulated a research framework about CI and IT flexibility. The research framework was leading for the mixed-methods research in chapter 8 and 9.

1.7. Research methods in detail

In this section a more extended explanation of the applied research methods is given, namely case studies and mixed-methods research.

Literature study. The literature study has been performed in periods between and after the case studies. In World Cat, Google Scholar and Web of Science search terms have been applied, such as Enterprise architecture, requirements, IT architecture, IT infrastructure in combination with terms such as software architecture, software development, modeling, conceptual modeling and technical implementation. High-level business terms and technical software terms have been combined in queries, such as strategic goals and IT solution architecture, to find literature about bridging the gap between software and high-level business requirements. The number of results of queries was small, mostly research domains such as model driven development and software development methods such as agile development surfaced. The studies of axiomatic design and conceptual independence were selected for this thesis as relevant.

Case studies. Case studies offer the opportunity to study the IT flexibility in-depth and collect information about the IT flexibility of an organization from different angles (Yin, 2009).

Selection of cases. The first three cases, chapters 2, 3, 4 and 6 have been selected based on the network of the author and related networks. Enterprise architects and IT architects of medium and large healthcare organizations were contacted. In the organizations the researchers were allowed to have discussions with a number of employees in the IT department by mediation of the IT architects.

The cases of open source Electronic HR software in chapter 7 have been searched for on the internet. Cases were selected based on an active community of users and state of the art software.

Cases in chapters 8 and 9 represent ten mental healthcare organizations that were contacted through the network of the author, the network of the openEHR supplier and the network organization in the Netherlands (De Nederlandse GGZ) consisting of 100 mental healthcare organizations. The selection included five cases that had implemented the design principle of CI and five cases that had not implemented CI. Here the researchers interviewed one or two IT professionals in every organization.

Data collection. Lee (Lee, 1989) describes four problems that have to be solved in case study research. In this section we explain how these four problems have been addressed.

First, the observations have to be carried out in a controlled way. In the first three cases in which enterprise architects and IT architects have been observed in their work of adapting existing IT infrastructures to new demands, we have been modeling the EA in the formal language of Ampersand. The first three cases each consisted of a period of about a year, during 2012-2016, in which the author proposed models which were discussed and iteratively adjusted according to feedback of the architects. Documents of IT architectures and reports of focus groups in the organization have been analyzed in the research. In the other 10 cases in mental healthcare, data were collected by a questionnaire and at the same time or afterwards semi-structured interviews were conducted that followed the order of items in the questionnaire (Schmidt, 2004). The interviews were transcribed and analyzed. All remarks of the architects and functional management in the interviews were noted and included in the data.

Deductions, replicability and generalizability. Lee argues (Lee, 1989) that deductions have to be made in a controlled way. For the three case studies all EA models in Ampersand have been compared to each other. The models and the comments and discussions of the enterprise and IT architects have been analyzed with content analysis (Mayring, 2004). The case studies showed that bottlenecks and difficulties existed when working with simplified EA models that had to be studied in other contexts.

In the next stage we analyzed and synthesized quantitative and qualitative data collected from a survey and semi-structured interviews with IT architects and functional managers in all organizations in the first semester of 2019. All organizations have responded to the same questionnaire and have participated in a similar semi-structured interview, thereby making the data comparable across organizations. For deduction we have relied on a mixed-methods design, see the next section. For replicability, we have found that although the sample consisted of small numbers of organizations, we stayed on alert for exceptional circumstances and factors in the cases that bias the conclusions.

Mixed-methods research. Mixed-methods research is defined by Creswell and Creswell (Creswell & Creswell, 2005) as "… a research design or methodology for collecting, analyzing and mixing both quantitative and qualitative data in a single study or series of studies in order to better understand research problems". Their method consists of collecting both quantitative and qualitative data and sequentially integrating quantitative and qualitative methods. We have collected quantitative data with a questionnaire asking respondents to rate items and qualitative data in semi-structured interviews about the same items.

Our approach is very similar to the mixed-methods approach of Dennis in exploring the adoption of use of group support systems (Dennis & Garfield, 2003). Dennis applies qualitative data analysis and adds a quantitative analysis to enhance understanding. In that process he uses statistical tests on small samples and combines quantitative and qualitative data to strengthen the results. He adds: "This mixed method of utilizing quantitative survey data with a small N to complement qualitative data has become accepted in IS research." (Dennis & Garfield, 2003). We have in a similar way combined a quantitative method for analyzing the data from the questionnaire (Nachar, 2008) and integrated the results with an analysis of the qualitative data from the semi-structured interviews (Mayring, 2004; Schmidt, 2004).

Quantitative data and interviews. The quantitative data have played a subordinate role in the 10 cases that we have studied. Our collected quantitative data consists of scores of the IT architect or IT functional manager on a questionnaire. The questionnaire is based on an empirically validated survey of Mikalef et al. (Mikalef, et al., 2016). Next to the quantitative data we have performed interviews with IT professionals in the organization.

Content analysis of qualitative data. In the period January to July 2019, 17 hours of interviews were recorded and transcribed. From July until November 2019, the text was split in 885 text fragments and these were summarized in English. The English statements that resulted were treated as qualitative data. Quantitative and qualitative data have been synthesized based on propositions or content analysis from July until November 2019 and again from January until July 2020.

1.8. Dissertation outline

Overview.
The papers in the dissertation view the IT change process from different angles. The high-level strategic view is represented by the enterprise architects, the mod-eling aspect by the models of flexible information systems, the developers view by open source code and the view of IT professionals in the organization by the multi-case study research in mental healthcare. These views demonstrate that a gap between the high-level view of IT infrastructure and the implementation of the software solution architecture still exists. Results in the papers suggest that the conceptual model of the healthcare professionals can be the bridge between high-level views and software developers' views. Conceptual models could aid in aligning the different views.

Integration of chapters and contribution.
With the objective to support healthcare organizations with managing new requirements for large, complex IT infrastructures, we have approached the IT infrastructures from different levels of abstraction. Therefore, the results are difficult to relate to a common position. Focusing on business requirements vertically from strategic goals to software terminology and methods has been difficult because information at this level of detail often could not be accessed. This study is contributing by opening up the white boxes and black boxes in software and to explain the structure of IS from a healthcare perspective.

Chapter 1. Introduction.
The initial focus of the research had been on designing and developing conceptual models and tooling for the practice of EA in healthcare organizations in the Netherlands. In this chapter the motivation for and the context of the research have been explained.

Chapter 2. Using ampersand in IT architecture.
Published as:
Tarenskeen, D., Bakker, R., & Joosten, S. (2013). *Using ampersand in IT architecture*. Paper presented at the 7th International Conference on Research and Practical Issues of Enterprise Information Systems CONFENIS 2013, Prague, Czech Republic.

Summary. IT architects in practice have a difficult task in extending and adapting enterprise IS. Mastering complexity is a first priority. In the study a conceptual model has been developed to help the enterprise architects with decisions about which parts of the IT infrastructure to change and which parts to maintain unaltered based on new functional requirements. The conceptual model has been formulated in the tooling of Ampersand. A prototype tool based on the conceptual model has been developed and evaluated by the enterprise architects.

Contribution to RQ1. A model has been designed for supporting enterprise architects in practice with their work of managing consistency and completeness of emerging architectures.

Chapter 3. Applying the V-model and axiomatic design in the Domain of IT Architecture Practice.
Published as:
Tarenskeen, D., Bakker, R., & Joosten, S. (2015). Applying the V Model and Axiomatic design in the Domain of IT Architecture Practice. *Procedia CIRP*, *34*, 263-268.

Summary. This chapter applies and discusses the principles of Axiomatic Design (AD) for changing IT architecture in healthcare. It presents three case studies positioned in the field of EA that explore how IT architects, as professionals, manage change and re-design the structure of the IT systems in line with strategic goals.

Contribution to RQ1. Enterprise and IT architects experiment with modeling IT infrastructures with the principles of AD to explore the possibilities of adapting existing IT architectures to new strategic goals.

Chapter 4. Reflections on Using an Architecture Model for Matching Existing Applica-tions to a Radical Business Requirements Change: a Case Study.
Published as:
Tarenskeen, D., Hoppenbrouwers, S., & van de Wetering, R. (2018). Reflections on Using an Architecture Model for Matching Existing Applications to a Radical Business Requirements Change: a Case Study. In *IFIP Working Conference on The Practice of Enterprise Modeling*, 383-393. Springer, Cham.

Summary. In this chapter a follow-up study on the EA model of chapter 2 is conducted. This EA model was meant for supporting enterprise architects with selecting the suitable applications for reuse. The way the organization manages next steps in EA is discussed with the enterprise architects. Reactions of other stakeholders and follow-up projects have been analyzed. This study reflects on the use of the model of chapter 2 in the hospital.

Contribution to RQ2. The application of a model for EA as designed in chapter 2 has been evaluated in practice by the author and enterprise architects involved in a follow-up study. A detailed study is made of the bottlenecks in follow-up projects for the EA project. Furthermore, the feasibility of implementation of decisions based on an analysis of new requirements for existing IT systems has been assessed.

Chapter 5. Conceptual Independence as an Architecture Pattern for Adaptable Systems.
Published as:
Tarenskeen, D. (2016, April). Conceptual Independence as an Architecture Pattern for Adaptable Systems. In *Proceedings of the 10th Travelling Conference on Pattern Languages of Programs,* 1-10. https://dl.acm.org/doi/10.1145/3022636.3022641

Summary. In the chapter a pattern for CI has been described. The pattern is based on literature study and demonstrates that flexibility in IS is feasible. It is illustrated with existing prototype systems and compared to patterns of model driven development.

Contribution to RQ3. In the chapter a software pattern for decoupling of conceptual models from application logic is described.

Chapter 6. IApplying Axiomatic design and Conceptual independence in the domain of IT systems.
Published as:
Tarenskeen, D., & Bakker, R. (2017). Applying Axiomatic design and Conceptual independence in the domain of IT systems. In *MATEC Web of Conferences,* 127(01006). EDP Sciences.

Summary. This chapter contributes to RQ2 and RQ3. For answering RQ2 the state of IT flexibility has been studied in three case studies. IT architects have been experimenting with models that represent the existing IT infrastructure where functional requirements are mapped to separate IT components. AD is set as a candidate for realizing flexible IT systems. In the study the principle of independence in AD has been examined in IT infrastructures of medium to large healthcare organizations. Registration of the mappings of functional requirements to independent software components has not been evident. The data collected in the organizations about demarcation lines in IT architecture have been interpreted to find reasons why AD has not been applied in large IT infrastructures in healthcare organizations.

Then for RQ3 a model has been designed for IS based on a literature study of AD and CI that would satisfy the requirements for AD by solving the issues found in practice with CI.

Contribution to RQ2. AD could offer a solution for enterprise architects and IT architects to improve flexibility in IT infrastructure. Hence we explore the degree to which IT infrastructures have applied AD. We study the degree of IT flexibility based on AD in real-life cases.

Contribution to RQ3. A model for IS infrastructure has been designed based on literature study of AD and CI, because AD claims flexibility of IS and CI offers a solution for problems and bottlenecks in IT systems in operation in practice.

Chapter 7. Unintended effects of dependencies in source code on the flexibility of IT in organizations.
Published as:
Tarenskeen, D., Van de Wetering, R., & Bakker, R. (2018). Unintended effects of dependencies in source code on the flexibility of IT in organizations. In *FedCSIS (Communication Papers)*, 87-94.

Summary. The hypothesis that the flexibility of business requirements is difficult in IT systems because of software dependencies has been stated before in research. This chapter focuses on the way domain knowledge is implemented in software code. The source code of three open source healthcare systems has been analyzed and conclusions about the degree of flexibility of these systems are drawn.

Contribution to RQ4. The chapter offers a further investigation of the interdependencies between conceptual models and application logic, focusing on how healthcare terminology is implemented in software in program code.

Chapter 8. Contribution of conceptual independence to IT infrastructure flexibility: The Case of openEHR.
Published as:
Tarenskeen, D., van de Wetering, R., Bakker, R., & Brinkkemper, S. (2020). The Contribution of Conceptual Independence to IT Infrastructure Flexibility: The Case of openEHR. *Health Policy and Technology, 9*(2), 233-244. doi: https://doi.org/10.1016/j.hlpt.2020.04.001

Summary. IT flexibility is needed to enable organizations to meet new demands. In this study we focus on Conceptual Independence (CI) because CI, as a design principle, can improve the adaptability of IS. It is examined if CI improves IS flexibility and, moreover, an impact of CI on the flexibility of the whole organization-wide IT infrastructure is searched for. With a mixed-methods approach data have been collected from multiple cases in healthcare organizations. Quantitative and qualitative data have been analyzed, synthesized and integrated. The implementation of the openEHR specification in EHR systems in healthcare has been researched as a proxy for CI and its effects on IT flexibility.

Contribution to RQ4. CI is a design principle, that shows how conceptual models can be separated from the application logic. It is claimed that in IS, where CI is implemented adaptive IS result and changing data models and reuse of functionality becomes easier. The impact of CI on the organization-wide IT infrastructure is examined.

Chapter 9. Investigating the Impact of Outsourcing on IT Flexibility: The Conceptual Independence Perspective.
Published as:
Tarenskeen, D., Van de Wetering, R., Bakker, R., & Brinkkemper, S. (2020). *Investigating the Impact of Outsourcing on IT Flexibility: The Conceptual Independence Perspective*. Submitted for publication.

Summary. In this chapter a more detailed investigation of qualitative data collected with semi-structured interviews in nine cases has been performed to find how CI sub-principles could aid in increasing IT flexibility. Then the study explains how the principles of CI have been applied in the organizations and how these result in a greater flexibility of the IT infrastructure as perceived by the IT professionals in the organizations. This study also explores the influence of outsourcing configurations on IT flexibility in the context of CI.

Contribution to RQ4. The study explores IT resources and capabilities that contribute to IT flexibility in organizations that have outsourced the IT development function. In the study organizations that have implemented CI and organizations that have not implemented CI have been compared.

Chapter 10. Conclusion and reflection.
In the last chapter a synthesis is made of the results and discussions of all chapters. Together the chapters clarify why research on IS and software structure is relevant for IT infrastructure flexibility. The results of the research have been evaluated and related to new research in the context of IS capabilities and enterprise architecture management (EAM) (Ahlemann, Legner, & Lux, 2020). A major contribution of the dissertation is that in EA the IS development is critical in achieving flexibility on an enterprise-wide level. Mechanisms behind the influence of IT resources such as IS software structure and IT outsourcing on IT flexibility are explained. The implementation of CI as a standard that decouples the conceptual models from the application logic leads to a rethinking of the structure of the IT infrastructure as a whole.

2

Using ampersand in IT architecture

IT architects in practice have a difficult task in extending and adapting enterprise information systems. Mastering complexity is a first priority. This is still more art than craftsmanship in spite of many years of academic research in IT architectures. We will explore by a series of case studies how to support IT architects in practice, with their work of improving consistency and completeness of emerging architectures. In this process, IT architects should be capable of discussing conflicting, incomplete, highly abstract or overly detailed requirements with stakeholders and developers, while keeping an overview of all priorities and constraints. In this research paper, we first discuss the research strategy, Design research combined with case study research, to address this question.

Next, we discuss Ampersand and its tool, a rule-based modeling language and engine developed in high-level IT consultancy. Ampersand seems promising for flexible modeling of IT architecture, and its rule engine can help IT architects manage consistency between requirements and IT solutions. An empirical exploration in developing the IT architecture for a new specialized child oncology hospital is used to check the premises of our research and the usefulness of Ampersand in this context.

Conclusions of this exploration have led to hypotheses for testing in further research. The hypotheses concern applying the Ampersand approach to effectively support IT architects' work.

This work was originally published as:

Tarenskeen, D., Bakker, R., & Joosten, S. (2013). *Using ampersand in IT architecture*. Paper presented at the 7th International Conference on Research and Practical Issues of Enterprise Information Systems CONFENIS 2013, Prague, Czech Republic.

2.1. Interest of society/Societal relevance

The health care sector in the Netherlands struggles with the costs and scale of developing and renewing information systems and information infrastructure. Because of changing insights in goals and new ways of working in care and cure, such as a more central position of the patient and their family, new high demands are made concerning the functionality and operation of information technology. In this study redesigning architecture is executed in an environment where an existing IT system and infrastructure is already functioning.

2.2. Contribution to science

2.2.1. IT Architecture and Enterprise architecture

IT research the term architecture is both "A formal description of a system, or a detailed plan of the system at component level, to guide its implementation (source: ISO/IEC 42010: 2007). " and "The structure of components, their inter-relationships, and the principles and guidelines governing their design and evolution over time." (The Open Group, 2011, p. 20). IT architecture can be applied to software applications in an Application Architecture (see par. 2.2) or to software of one or more integrated systems in Software architecture. More and more, authors position IT architecture on Enterprise level (system of systems, including software middleware) and view the software and hardware as a whole in a supporting application landscape for the enterprise. Enterprise architecture is defined as: "fundamental concepts or properties of a system in its environment, embodied in elements, relationships and in the principles of its design and evolution." (Lankhorst, 2013).

Existing IT systems in large organisations are complex systems. User organisations perceive the need for comprehensive overview and insight in their architecture. Seltveit phrases the term "systematic complexity reduction", when discussing information system models. He focuses on comprehensible views, by customized filtering of the information. He notices the dual nature of IS models used for understanding and for designing systems (Seltveit, 1996). Leitner describes measurement of complexity of models (Leitner, Weiß et al. 2012).

Complexity, however, is not systematically reduced. In standard works for architects in practice published by the Software Engineering Institute, Bass, for one, advises use of checklists for guarding quality attributes when designing IT architecture. These checklists do not integrate into a clear direction for making decisions, and they do not reduce complexity for architects. "You as an architect must keep in mind the constraints given to you and arrange the composition so that it will accommodate those constraints" (Bass, Clements et al. 2012). Clements advices on a thorough documentation of architecture for communication of structure and of choices made (Clements, Bachmann et al. 2011). Views as proposed by Kruchten are another way to reduce complexity of the domain without losing the essential relations (Kruchten, 1995). The focus here is on software architecture.

In the last decade, however, a whole new approach for addressing complexity has been proposed, because lines of research did not lead to satisfying results. In Rethinking architectural decisions, Tyree and Akerman re-emphasized the rationale behind the decisions. (Tyree and Akerman, 2005). Not decisions, but architecture principles are formulated by Greefhorst as central concepts. They should lead to restrictions for system design (Greefhorst, 2011). Rethinking the structure of the IT architecture took place, after the diagnosis has been made by Hoppenbrouwers and Greefhorst that terminology and method in the IT architecture domain had led to unmanageable diversity (Hoppenbrouwers, Proper et al. 2005; Greefhorst, Koning et al. 2006).

Furthermore, IT architects have collaborated in the Open group by developing an overall method for Architecture development with TOGAF, where all relevant architecture methods are included in an ordered way (The Open Group, 2011). This way, IT architects could benefit from a collection of “best practices”. The terminology or ontology of TOGAF is based on NEN/ISO 1471 and ISO/NEN 42010:2011. “Enterprise principles provide a basis for decision-making throughout an enterprise, and inform how the organization sets about fulfilling its mission”, are however beyond the scope of the architecture, because they belong to the domain of IT Governance (The Open Group, 2011, p. 235, p. 593).

In the work of Lankhorst a foundation is laid for enterprise architecture modeling based on detailed relations between Business processes and supporting applications. This, however, leads to so many elements and relations that tooling, such as ArchiMate, is recommended (Lankhorst 2013, p. 245). Different tiers are included in the ArchiMate tool for requirements, low level business processes and technology. Later, a way to trace back to high level business goals and strategy in the system is added with Motivation extension. This addition has not yet been fully integrated in the rest of the system. We have not realised a tool that is capable of managing IT through a set of integrated views of the enterprise (Lankhorst, 2013, p. 266/267).

The Ampersand approach that is applied here attempts to simplify the number of concepts and relations by adding high level enterprise goals and strategies to low level requirements that are fully integrated in the model.

2.2.2. Existing Application Architecture

Application architecture is defined as a “description of the structure and interaction of the applications as groups of capabilities that provide key business functions and manage the data assets.”(The Open Group, 2011, p. 20).

An existing application architecture adds extra complexity to the number of factors in IT architecture, recent research in Health care systems therefore, emphasizes the importance of an analysis of existing IT infrastructure (Chiu, Kohli et al. 2012; Larson, Hatcliff et al. 2012).

A recent multiple case study in complexity of approaches in redesigning systems, concludes that building the design bottom up from the existing infrastructure leads to reducing complexity. This, in contrast with projects where the architecture was developed top down, or middle out, where complexity increased (Zickert and Beck 2012).

2.3. Method

2.3.1. Design research and case studies

For research the case study method in combination with design research is proposed (Hevner, March et al. 2004; Hevner and Chatterjee 2010; Yin, 2002). The case study method will be used for observation and inductive building of theory. Later case studies will be extended with testing of hypotheses. The research consists of a series of case studies in which IT architects in practice are supported in their work of designing and documenting architectures.

Design research is used as a method for studying created artifacts, such as models, designs and tooling, that are meant to support the practice of IT architects. Hevner and Chatterjee (2010) argue that design research can lead to new knowledge in the domain concerned, when relating the design to results from scientific research. The IT artifacts and their characteristics must be precisely defined. Furthermore, the relation of design research to a kernel theory is recommended. In this case, relation mathematics for checking consistency is applied. Design knowledge can be stored in meta-artifacts, in this research it means that knowledge for the IT architecture practice will be embedded in the artifacts resulting from the research: they can be analysed as meta-models and prototypes for a class of Information systems (Hevner and Chatterjee, 2010, p. 51). This knowledge can improve the practice studied.

We have searched for a tool for constructing artifacts, such as conceptual models and prototypes. This tool needed to function in a practical context, therefore we required a flexible tool, that could create ontologies based on terminology of IT architects and that did not add complexity in the form of pre-existing concepts and mandatory relations. In the case study the ontology had to be defined by the architects, and not imposed on them by the tool. The concepts selected by architects could thus be studied.

Summarized, a tool was required that could be used to build conceptual models and check rules, preferably stated in (structured) natural language. The selected tool added prototype-generating capabilities. In this way feedback from IT architects could be collected directly and with less ambiguity.

In the exploratory study performed, the inductive process is followed, with the objective to formulate hypotheses for follow up research. In later research, both inductive and deductive processes will be included, see Cycle of theory building (Hevner and Chatterjee, p. 34). Models and artifacts in all case studies will be subject to qualitative analysis.

2.3.2. Ampersand approach

Recently, the Ampersand approach was introduced by Joosten (2007) in which an ontology written by an analyst is transformed into an information system by means of a software generator. Early experience in industry with Ampersand shows promising results: a system can be built with relatively little effort just by specifying an ontology (Joosten, 2007). We expected that Ampersand could directly help to organize and model IT architectures, thus contributing to a more professional approach of architecture.

This approach creates a data space from a user specified ontology, and implements it by means of the rules that govern the ontology. The generated software is a prototype web- application, in which the user can alter the data within the limitations set by constraints. This mechanism has proved useful for designing business process management systems by a large system integrator in the Netherlands, merely by formalizing business rules. The results have inspired us to implement the bottom-up collection of architectural information and evaluate the data with specific rules. By using the Ampersand mechanism to design an architecture repository by architecture relations and constraints only, Ampersand can also generate the artifacts necessary for the Design research.

We expected wins when using Ampersand in the IT architecture domain. First, we aimed at complexity reduction. Secondly, we assessed that consistency checks would be useful.

The Ampersand approach as proposed in this paper, is defined in the following way:

- Ampersand is used as a business rules approach for Enterprise architecture. The rules are formulated in relation algebra and are based on research in the domain of business rules (Joosten, 2007)
- Ampersand defines the architecture in concepts and relations
- the stakeholders are responsible for deciding on terminology and on definitions of relations in their own language
- methodical use of relation algebra is used for defining the conceptual model in Ampersand with relations and business rules in the IT architecture domain (Michels, Joosten et al. 2011)
- the Ampersand tool is used to perform consistency checks on relations using a form of relation algebra to accomplish this. (Michels, Joosten et al. 2011).

2.4. Exploratory study

2.4.1. Description Princess Máxima Center (November 2012-April 2013)

The exploratory case study is undertaken in Princess Máxima Center for Pediatric Oncology with the objective to observe the work of IT architects in practice. Design models were created by the IT architect and the researcher and were reviewed and evaluated by them in three iterations.

The case consists of the development of an IT architecture for a new hospital specialised in child oncology. The existing integrated information system of a nearby academic hospital can be re-used. The existing infrastructure already services an organisation of about 1000 beds and 11.000 employees.[1] The ambition of the new hospital is to innovate the care processes, and to integrate care and research in the field of pediatric oncology. Primary treatment and all research will be centralised in Utrecht. Less complex parts of treatment will be distributed over The Netherlands in shared care centres. This case has been selected because of the high ambition of the project. It aspired to restructure the existing IT infrastructure.

The complexity of requirements and IT-applications were represented in a conceptual model in Ampersand. A prototype (Android app) was developed by students and in this prototype "violations" of consistency checks could be viewed. Eleven groups of about 5 stakeholders and experts each discussed the information model for a specific functional domain. The results from these discussions have been used by the practicing IT architect and project manager to formulate the data for the conceptual model. The model and all input were formulated in their terminology.

Data was checked by the Business rule engine. Afterwards the results were reported back to stakeholders as "violations of the rules". Their evaluations were noted.

2.5. Results

2.5.1. Conceptual model

In this paragraph the conceptual model is described. The model consists of relations, concepts and rules.

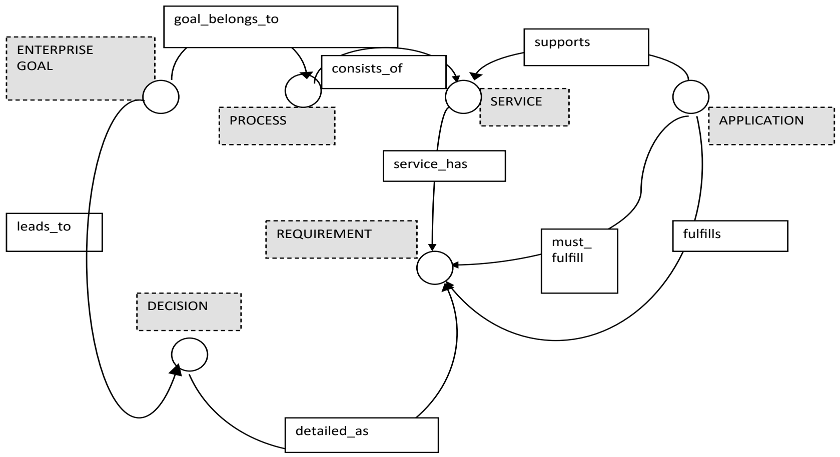


Fig 2.1. Conceptual model Princess Máxima Center

The relations between "High level enterprise goals" and "(Cure and Care) Processes" and "Services" are defined. Processes are derived from existing Hospital reference ar-

1 http://www.umcutrecht.nl/overumcutrecht/Kengetallen-2011.htm

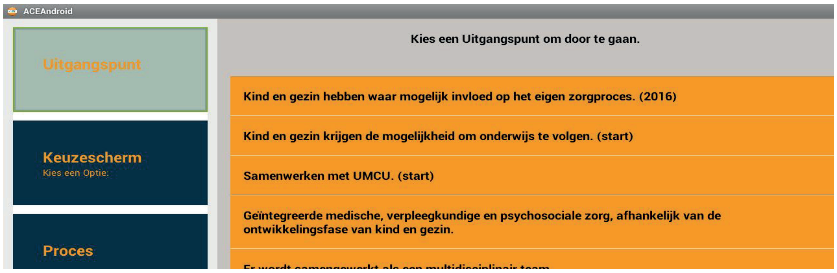
chitectures. Services are composite (IT-)functions of system or modules. Requirements are formulated by IT architects. Applications have two relations with Requirements: a must_fulfill relation and a fulfills relation.
Example:

High level Enterprise goal: "Child and family can influence the cure process if possible" is related to processes such as: "Health Care Patient portal", "Medication, Pharmacist, Cytostatics". These lead to Services as: "Facilitate education", "Give information", "Communicate with caretakers and peers", "Support care process", … When zooming into "Support care process" one can see 4 existing applications: "Patient dossier management module", "Documentation module", "Health care portal", "Portal for health care employees".

Input of the model consisted of:

- 27 High level requirements/Business goals (Enterprise principles, TOGAF)
- 12 processes, 236 services, 90 applications, 133 specific requirements, more than 2000 relations
- 10 Rules (of which 4 Consistency checks).

2.5.2. User interface screen of Android application



Kies een Uitgangspunt om door te gaan.

Kind en gezin hebben waar mogelijk invloed op het eigen zorgproces. (2016)

Kind en gezin krijgen de mogelijkheid om onderwijs te volgen. (start)

Samenwerken met UMCU. (start)

Geïntegreerde medische, verpleegkundige en psychosociale zorg, afhankelijk van de ontwikkelingsfase van kind en gezin.

Fig 2.2. Example User interface with High level business goals

In a screen of an Android app the High level enterprise goals are presented in a list; clicking on one of them leads to related Processes, Services, Applications and Requirements.

2.5.3. Consistency checks

For every Requirement belonging to a Service or an Application it was tested if there was an Application that fulfilled the requirement. If not a violation was notified, violations were shown on screen (not shown here).

2.6. Discussion

2.6.1. Questionnaire IT Architects

The questionnaire consisted of 11 closed questions and a requests for comments. The IT architect and IT project manager, who both developed and evaluated the IT architecture have responded to these questions and have positively evaluated the conceptual model for representing the complexity and the application portfolio. They agree on its value for discussion with stakeholders. They agree on possibilities that the tool offers for checking consistency of applications to requirements. However, further research is necessary. For instance, the terminology of the model should be critically reviewed. Crucial is the guarantee that input data is complete, and that functional relations are visually presented. The IT project manager emphasized the importance of a flexible way to adapt information in the model.

2.6.2. Artifacts made by Ampersand can reduce complexity and support traceability

We expected to receive wins from this approach. When evaluating the results the wins can be classified as complexity reduction and/or traceability:

- Complexity reduction. By letting Ampersand in prototypes do the checking on consistency, the IT architect can focus on content of the Enterprise architecture.
- Complexity reduction. By using the conceptual modeling in Ampersand, all facts and expressions could be formulated in binary relational form using Excel spread sheets.
- Traceability. By analysing the conceptual model in Ampersand, we found that the respondents prefer to trace all facts and expressions back to high level goals and strategies.

More objectively, it can be observed that the conceptual models in the case study, are more simplified than the models that are recommended in the ADM of TOGAF. The number of concepts is smaller than the number in the terminology of TOGAF. The concepts in the case study are formulated on a higher level of business goals than in TOGAF. The respondents indicate positive effects of using the artifacts in complexity reduction and traceability. Focusing only on *binary* relations proved a simplification.

2.6.3. Ampersand can signal Fit and Gap

We further expected, that by letting Ampersand use Business rules to compute violations of the model, the stakeholders could more clearly notice gaps in requirement fulfilment. The IT architect referred to the “violations” as Fit and Gap of applications. The conceptual models used here contain relations on a level that is not included in the method of TOGAF (The Open Group, 2011). In comparing the Ampersand tool to ArchiMate we find that ArchiMate signals Fit and Gap between two architectures, the Base architecture and the Target architecture (The Open Group, 2012), while we signal a Fit and Gap of applications to requirements. In the follow up the approach of the direct mapping of high level business goals to applications, has to prove itself in communication to and with the technical experts and developers.

2.7. Conclusion

The results indicate that IT architecting in practice need methods for complexity reduction. Respondents recommend further research. The results of the exploratory case thus confirm a possible win from this approach. The following hypotheses will be examined:

1. *The Ampersand approach as described above, effectively supports IT architects in practice, who are (re)designing a system based on requirements, during the design phase of the enterprise architecture.*
2. *The IT architects can be effectively supported by models and tools, that reduce complexity, facilitate traceability and provide fit and gap analysis.*

In the next stage we will also explore whether these statements are valid not only for the stage of designing the IT architecture, but also for the implementation of the design. Then, research questions will be examined in a more systematic way than this first exploratory case study allowed. The focus on *enterprise* architecture can be used to improve theory in Information system development.

3 Applying the V Model and Axiomatic Design in the Domain of IT Architecture Practice

This paper applies and discusses the principles of Axiomatic Design for changing IT architecture in health care. It presents three case studies positioned in the field of Enterprise architecture that explore how IT architects, as professionals, manage change and re-design the structure of the IT systems in line with strategic goals. The research approach was to use a light modelling tool, Ampersand, for modelling the Enterprise architecture. Two types of models stand out: Type 1 Strategic IT models in which higher strategic goals are related to requirements for applications and Type 2 Technical management of systems models in which technical risks and risk of system failure in the current IT infrastructure were modelled. To bridge the views of different IT experts in the organization this work uses the customer domain, the functional domain and the physical domain from Axiomatic Design in an extended example in the paper. The V Model is used to bridge the models, and then it is extended with Axiomatic Design principles.

3.1. Context and original research questions

3.1.1. Research question and case selection

Our original research question was: How can we improve the practice of the IT architect in health care with IT architecture models? First, three exploratory case studies were planned to provide one or more hypotheses about the models of IT architecture. A case study is seen as a mature research method for building theories in this field (Eisenhardt, 1989; Lankhorst, 2013; Yin, 2002). The cases are examples of paradigmatic cases, because they provide typical information about the IT architecture practice, and extreme cases, because the context is one in which IT architecture is new (Flyvbjerg, 2006).
In the cases IT architects explore the IT application landscape in order to decide which

This work was originally published as:
Tarenskeen, D., Bakker, R., & Joosten, S. (2015). Applying the V Model and Axiomatic design in the Domain of IT Architecture Practice. *Procedia CIRP*, *34*, 263-268.

parts of the landscape have to be changed or replaced. This is a real-life goal for IT architects, as an existing IT landscape always exists. In the cases, changes in the IT architecture were needed because of changing strategic goals of the organisations. In all cases, the (TheOpenGroup, 2011) starting point has been the document of strategic goals of the health care organisation. The models had to take these goals into account while applying Enterprise architecture scope on the IT architectures (Lankhorst, 2013).

The original idea was to start from research on IT architecture and methods that are widely used such as TOGAF and Archimate, methods for developing and detailing Enterprise architectures. We planned to use models from these methods that would then be input for discussions with stakeholders (Lankhorst, 2013; TheOpenGroup, 2011). However, this is not how the case studies were conducted. The IT architects involved had their own way of modelling and strived to get a grip on the complexity of the existing application landscape. Our research followed the systematic approach that IT experts used.

Although concepts and relations have a similarity to concepts in TOGAF and Archimate, they were not explicitly applied in these models. The IT architects used a proprietary reference model for hospitals based on Ross and Weil (Ross, Weill, & Robertson, 2006) and the in the Netherlands broadly accepted 'reference model of hospitals' of Nictiz 2012-2014 (Fischer & Smeele, 2014).

3.1.2. Modelling tool

We have used a modelling approach called Ampersand, because of its lean model and its possibilities to formulate rules based on relation algebra for checking consistency of data. Ampersand uses concepts, relations and business rules in a model. These terms in the model are defined in the Ampersand metamodel. Ampersand includes a tool for checking the consistency of the IT architecture data with the formulated business rules. The flexible modelling aspect was especially needed because the models changed often during discussions. The business rules helped to analyse the underlying goals (G. Michels, S. Joosten, J. Woude, & S. Joosten, 2011; G. Michels, S. Joosten, J. v. d. Woude, & S. Joosten, 2011).

3.1.3. Data collected

The resulting data consists of models in different versions, reports and mails with discussions between IT architects, IT senior experts, an Information architect and one of the researchers. After each case an evaluation session of the project and its concrete benefits for the organisation was held. In case 1: information is based upon models with input from an IT architect and an IT project manager. We had access to reports of 11 theme groups of medical specialists and hospital employees in which requirements where discussed. We had access to drafts for the IT architecture of the future hospital and the technical infrastructure. We had access to end reports of the preparation phase and to a specific plan for adapting a selected application, based on the elicited requirements. Models have been filled with data and the rule engine of Ampersand has been applied for formally testing rules.

In case 2: Information is based on models with input from an IT architect and a senior IT expert from the IT department. Models have been filled with data and the Ampersand rule engine is applied for formally testing rules. We had access to documents describing the strategic goals of the organisation. We developed an instrument for assessing technical risks for the existing applications in cooperation with both IT experts. We had access to an advice of the IT department for adding a data service bus to the architecture for accessing distributed data. During this case, interviews were held with relevant IT experts in the field of technical management of systems.

In case 3: Different versions of the model are discussed with two IT architects in the hospital. The model is currently under construction. Models have been filled with data and the Ampersand rule engine has been applied for formally testing rules. We have access to the drafts and definitive goal architecture document and drafts for the IT architecture diagrams.

3.2. Axiomatic Design

3.2.1. Overview of Axiomatic Design Theory

The ideas of Axiomatic Design (AD) originate from industrial production and industrial systems, but they are relevant for software and hardware systems. The design method bridges different domains when describing the design of a suitable system. Two axioms stand at the basis of this method (Kulak, Cebi, & Kahraman, 2010; Suh, 2001). It can be mathematically demonstrated that designs following these axioms are better designs, in the sense of less complexity and less costs (Malaek, Mollajan, Ghorbani, & Sharahi, 2015; Togay, Dogru, & Tanik, 2008).

AD assumes that designing systems requires input from different domains. The domains that are introduced are: the customer domain, where customer needs are elicited, the functional domain, in which functional requirements are positioned, the physical domain for describing the design parameters of the system and the process domain that complements the foregoing domains with information for the process of manufacturing the system.

The first axiom states that when a design is made all Functional Requirements (FRs) must be formulated on a fundamental level, in such a way that they are independent from each other. The FRs are later mapped to its corresponding Design Parameters (DPs). A DP describes a property or characteristic of the system. It is expected to fulfil the related FR.

The second axiom states that for every DP we should minimize the information content (Suh, 2001). It states that if we can choose between alternatives for a DP, choosing the DP with the highest probability of success leads to the best design. As low information content minimizes complexity, and thus contributes to a better chance of implementing the DP. Theoretically the information function is defined with a probability function. After that the probability of fulfilling the requirement (FR) with the realized DP is estimated. In software engineering we found in accordance with the comment of Suh that often there exists no observable Design Range where a DP fulfils the FR, because the DP either fulfils the FR or it does not (Suh, 2001).

Thus, it is not possible to estimate the probability of fulfilling the FR with the DP. But an estimation of the probability of delivering the DP can be made. One can for instance compare difficulties in realizing the DPs with different software constructing processes and tools. The difficulty then depends on the needed knowledge for realising the DP. More often than not, knowledge of different tools and libraries is necessary. This increases the information content of the DP. In practice one often refers to examples of DPs as "proven technology", when these are demonstrable and operational in other organisations, and thus give an indication of its probability of successful realization. In all case studies definitive decisions about DPs were made based on "proven technology". This was not explicitly examined, but was found in the cases.

3.2.2. Axiomatic Design in this study

The case studies were exploratory and resulted in different models that conformed to the requirements of the designers. However, the models were fragmented and each showed different parts of the reality of the different IT experts. We needed a Design Matrix (DM) to combine the models, and AD for assessing the quality of the models.

We have first applied the V Model in this context to map the business requirements to system functionalities for the existing IT application landscapes. The V Model is explained in section 3.4.

We then transformed the table in the V Model to a DM, we replaced business requirements with FRs and system functionalities with DPs. Then we placed the DM at the bottom of the V Model for communicating it to the IT architects and show the connection to the existing IT landscape. We expect to improve the design withAD.

3.3. Cases

3.3.1. Three case studies

The case organisations were all in the process of reorganizing the core business, the care process, due to changes in governmental regulations, changing vision on health care and a more central role of the patient in the interaction with medical personnel. They have an IT infrastructure in operation that functions sufficiently for the current requirements. The board of directors in all three cases wanted to re-evaluate the IT systems in the light of new requirements.

The researchers collaborated with the IT architects or with the IT department. The models were developed iteratively. Models that "were interesting" led to more versions, sometimes up to 10 versions. Models that had no relevance to the practice of the architects were abandoned after one or two versions. The rules for checking consistency of models were formulated in natural language by the IT architects and formalized by the researchers.

Table 3.1. summarizes similarities and differences in the context of the case studies.

Table 3.1. Context in different case studies.

Characteristics	**Case 1**	**Case 2**	**Case 3**
Changing Strategic vision	x	x	x
Evaluation of IT infrastructure	x	x	x
Need for new design of IT infrastructure	x		
Need for replacing parts of the IT infrastructure		x	
Need for extension of IT infrastructure			x
Contact IT architects	x		x
Contact IT experts from IT departments		x	x

3.3.2. Health care organisations in case studies

3.3.2.1. Case study 1: National hospital for child oncology

The reason for developing the architecture was the establishment of a new national hospital for child oncology, that would integrate laboratory functions and treatment, integrate knowledge from research in this area and treatment and coordinate care centres across regions in the Netherlands. Information technology that existed in an academic hospital nearby would be reused where possible. The IT architecture model, that was eventually selected by the IT architects can be seen as a simplified version of an enterprise architecture in which strategic goals were detailed in requirements for applications.

3.3.2.2. Case study 2: Medium sized regional centre in health care

The case organisation in health care functioned mainly as a care organisation and provided home care and care for the elderly. It had 5000 employees and about 3000 volunteers at the time of study. The IT architect and IT expert started with formulating strategic goals, because the organisation was in the process of changing its health care vision and strategy. The ultimate goal however, was to judge if the current system was up to the required changes and could be maintained in its current form. The board of directors was planning to update and replace parts of the system that were not functioning according to state of the art requirements.

3.3.2.3. Case study 3: Ongoing case: medium sized hospital

In the third case study a medium sized hospital in the Netherlands is being studied. The hospital is in the last stage of changing its paper-based operations to digital operations in the health care processes. The goal of the IT architects is to deliver a consistent vision of the changes in health care services as foreseen and the specified supporting IT functionality.

3.4. Findings

The first two case studies resulted in five distinct architecture models, these have been iteratively developed in a total of 30 versions. Fig. 3.1. and 3.2. show examples of the two types. We describe the two types in the next section.

3.4.1. Types of IT Architecture models

The five architecture models can be ordered in 2 types of models: one type in which strategic goals are leading and one type in which the technical management of system components was a priority. It makes a difference whether the IT experts apply an enterprise level point of view or a technical management point of view. Analysis of the rationale of involved parties point out that underlying goals for constructing the architecture models differ. These goals determine the ordering and structure of concepts. The difference is so fundamental that connecting different models is difficult or leads to a meaningless mapping between the elements of different models. Due to the complexity and chaotic structure of the relations between the models, checking whether technical systems sufficiently support strategic business goals is not possible with these models.

Type 1 Strategic IT models show models in which higher strategic goals are related to requirements for applications and decisions about the IT architecture. See example in Fig. 3.1. The architects wanted an overview the information that would abstract from the complexity numerous elements and relations between them. The goal of the model was to present a view of "Fit and Gap" of existing applications. Business rules in Ampersand are applied for signalling gaps.

The mapping of requirements to applications proved to be a bottleneck because the information about the internal structure of applications was excessive and detailed. The project was cancelled because of a decision that in the first phase of implementation all existing IT infrastructure of a nearby academic hospital would be used "as is". Adaptations for the IT architecture were delayed until 2015.

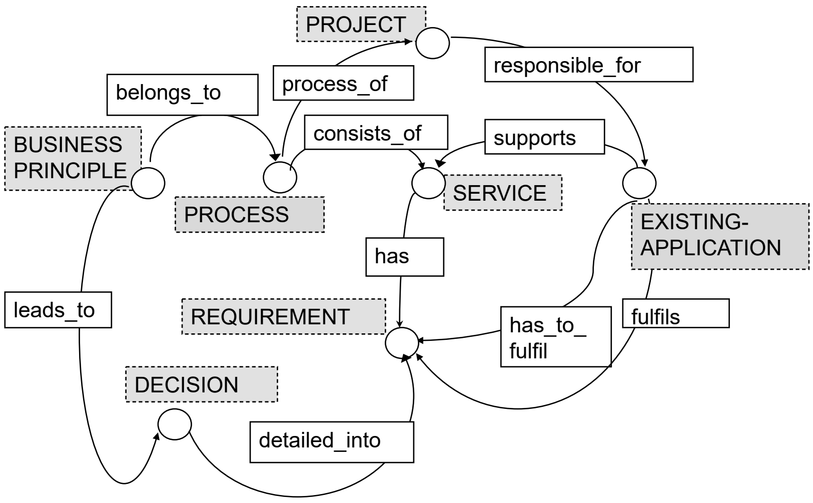


Fig. 3.1. Example of model of type 1 Strategic IT

Type 2 Technical management of systems models consist of 3 models in which technical risks and risk of system failure in the current IT infrastructure were modelled. In terms of AD theory, Type 1 models show the customer needs and the functional requirements that are derived from them. Type 2 models represent an image of the runtime system with system characteristics and dependencies.

For type 2 models the researchers developed an instrument to assess the risks of certain applications. The instrument is based on theory and adapted in interviews with the IT experts of the organisation. It was based on Andreou (Andreou & Tziakouris, 2007), NEN-ISO Standard Riskmanagement (InternationalElectrotechnical-Commission, 2009) and Lock and Sommerville (Lock & Sommerville, 2010). This instrument was tested in interviews with 2 other IT experts from the IT department.

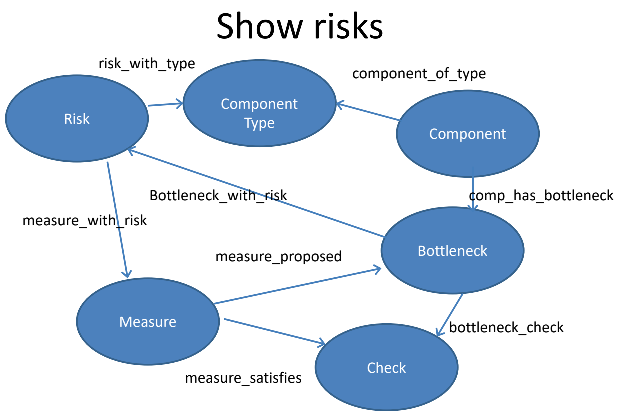


Fig. 3.2 Example of model of type 2 Technical management

A remarkable finding in this case study was that after the technical models were constructed, relations to the strategic goals of the organisation were not added to the models. Showing that these IT experts did not include the strategic goals in their models, and had no reference to them from the models that they considered meaningful, hereby confirming that the customer domain was not relevant in their models.

3.4.2. Matrix in V Model

In the context of case 3 we expected a multitude of relations from strategic goals to IT systems that would be difficult to interpret. Therefore the researchers developed a model in which the business functions and IT system functionality would be defined on a detailed level. Thedetails were used to demonstrate the possibility to bridge the two types of models. We had not yet applied AD in this stage of research.

We constructed a V Model that showed the relation between two different points of view in IT architecture on a deeper level of detail. We used zigzagging between domains to get to this level. This V Model can be seen in fig. 3.3. The specification of strategic goals to sub goals and requirements stemming from business can be found on the left side of the V. The right side of the V is used for mapping system functionalities upwards to a technological IT architecture. The latter is a representation of the runtime systems that are inuse in the organisation. When comparing the V Model with the Design Method of AD the left leg shows the Customer needs mapped to the FRs in the functional domain. Since IT architects work with these two domains, they can model them in one line.

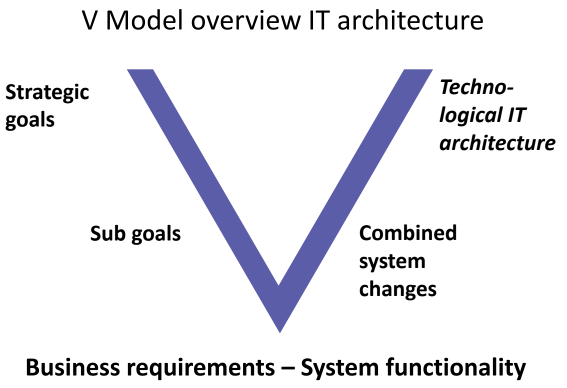

V Model overview IT architecture
Strategic goals
Sub goals
Techno-logical IT architecture
Combined system changes
Business requirements – System functionality

Fig. 3.3. V Model for overview of IT Architecture.

The right leg shows the mapping of different levels in the physical domain, where DPs on lower levels are mapped onto DPs in higher levels on the right leg. This line does not combine two domains in the AD. It shows only the running/runtime system and the required changes. We have used this diagram for communication purposes with IT architects and found a natural understanding. Even though this V Model was grounded on a deeper level on zigzagging from functional domain to the physical domain. The process domain is out of scope, because designing of program coding is not part of the models. The runtime system does not consists of program modules but of different system components that can be started separately and communicate during runtime.

An illustration of the matrix at the bottom of the V Model in case 3 can be found in table 3.3., without AD. In Table 3.2. four sub goals are shown. These sub goals belong to the strategic goal of increasing regional collaboration.

Table 3.2. Sub goals left side V Model

	Sub goals of Increasing regional collaboration
1	Improve ways for requesting medical examination by first-or primary care
2	More and better ways to judge examination results
3	Better and more direct contact between first-or primary care and hospital
4	Permissions for first-or primary care to access EHR Electronic Health Records or EMR Electronic Medical Records of their own patients in hospital EMR

In Table 3.3. the matrix at the bottom of the V is shown. In this table for two out of the four sub goals the business requirements are formulated and determined. They are linked one by one to required system functionality. AD is not yet applied here.

Table 3.3. Matrix V Model

	Business requirement	**System functionality**
1.1	Electronic request for lab examination by GP (general practitioner)	Access and interoperability for electronic requests for lab system hospital
1.2	Electronic request for lab examination by GP	Processing electronic requests by lab system
1.3	Electronic request for radiology examination by GP	Access and interoperability for electronic requests for radiology system hospital
1.4	Electronic request for radiology examination by GP	Processing electronic requests by radiology system
1.5	Electronic request for medical examination by GP	Access and interoperability for electronic requests for medical examination system hospital
1.6	Electronic request for medical examination by GP	Processing electronic requests by medical examination system
1.7	Service catalogues requests by first-or primary care	Access and views for electronic requests for service catalogues
2.1	Electronic consultation or automatic reception of images in required quality	Access and views for Image processing system PACS2 hospital
2.2	Electronic consultation or automatic reception of results of examinations by specialists of requested examinations	Access and views for requesting results lab
2.3	Electronic consultation or automatic reception of results of examinations by specialists of requested examinations	Access and views for requesting results radiology
2.4	Electronic consultation or automatic reception of results of examinations by specialists of requested examinations	Access and views for requesting results medical examination

3.4.3. Example of Axiomatic Design

We rewrite the matrix business requirements 1.1 and 1.2 in the form of a DM in the meaning of AD, by setting a one on one mapping from FRs to DPs and claim that the DPs can be fixed, in such a way that DPs that are on the left of other DPs are not changed.

In the example in table 3.4. is seen that one of the FRs, FR3, is referencing 3 DPs. The FRs are:

1. Access electronic request in lab system
2. Process electronic request in lab system
3. Handle electronic request for lab examination by General practitioners

They can be combined together in the following way with DPs. The DPs are:

1. Security system for electronic requests for lab system hospital
2. API Application Programming interface for electronic requests by lab system
3. Service for lab requests General practitioners

The first two DPs will be realized in the backend of the server program, both DPs are used by the third DP that combines them to provide the service needed by the user for handling the electronic request.

Table 3.4. Example Design matrix case 3.

	FR	DP1	DP2	DP3
1	Access electronic request in lab system	x		
2	Process electronic request in lab system		x	
3	Handle electronic request for lab examination by GP (general practitioner)	x	x	x

This can only be a responsible design according to AD if the first two DPs are "fixed" (Suh, 2001). In software the fixation is realized by encapsulation of the functionality that can be called through a specific, defined and fixed "interface" or definite system function call. Underlying functionality can then be changed without affecting the calling system functions as long as the "interface" is not changed. This practice is common practice in Software engineering and is regarded as good design (Larman, 2005).

By fixing the underlying DPs numbers DP1 and DP2, the design is triangular and decoupled.

3.5. Discussion

3.5.1. Goal oriented character of models

We presume that underlying goals of IT experts are the reason for different structure and ordering of models, in the cases.

3.5.1.1. Strategic IT models of IT architects

The IT architects have a task in making sure that IT is up to date and can support the organisation as a whole, the health care processes and its supporting processes. They make decisions about the structure of the IT application infrastructure and perform a fit and gap analysis They advise which parts have to be changed or adapted for needed functionality. The traditional idea of Business IT alignment is the responsibility of the IT architects (Henderson & Venkatraman, 1993).

3.5.1.2. Technical management models of IT operations and IT departments

The second group of IT experts work in the IT department and manages the IT systems.

The IT departments have a responsibility for the totality of the IT infrastructure. They guarantee to operate a running, stable, and secure system. They adapt the applications when a detailed request with permissions from higher management is received. They make decisions about rules for users and constraints for adaptation in order to guarantee stability and security. The IT department has the most extended knowledge of applications, their functionality and the options for storage and adaptations that are activated in case of system failure. According to Bass et al. (Bass, Clements, & Kazman, 2012) software architecture contributes to the quality attributes of software.

3.5.2. Bridging models with Axiomatic Design

We have applied the V Model in this context to map the business requirements to system functionalities for the existing IT application landscapes. We later used the principles of AD to replace business requirements with FRs and system functionalities to DPs. We have placed the DM at the bottom of the V Model for communicating it to the IT architects and show the connection to the existing IT landscape.

The DPs are designed in such a way that ideally every FR can be satisfied with one or more DPs in an independent way. Our goal to show traceability in design can thus be achieved.

Another advantage of applying AD for design is the opportunity to change FRs without compromising other functions, because in AD the FRs are independent of each other. A different approach, Model driven engineering, has been taken by Verelst (Verelst, 2005) in his research on Normalized systems, where software systems are generated in such a way that every system function is independent of other system functions. This approach is not suitable in our cases because a replacement of the infrastructure as a whole is not feasible in this complex, operative context.

3.6. Conclusions

We have found two types of models of IT architecture that both contain relevant information for evaluating the IT infrastructure in supporting the goals of the organisation. Bridging the models is necessary, because IT architects in the study need to relate their strategic goals to the IT systems in order to check traceability and soundness of design of the IT architecture. The gap can be overcome by using the DM of Axiomatic Design. For visualizing the required changes to the existing IT architecture the V Model can be used. Axiom 1 is compatible with best practices in IT development. Axiom 2 needs further research.

3.7. Future research

Our research will proceed with the study of bridging different IT Architecture models in an organisation with the V Model and AD. Priority will be given to the study of effects of applying AD on the practice of IT architecture.

Acknowledgements

The research is part of PhD research of D.Tarenskeen and was performed in the HAN University of AppliedSciences, Arnhem, The Netherlands and the Open University, Heerlen, The Netherlands

4 Reflections on Using an Architecture Model for Matching Existing Applications to a Radical Business Requirements Change: a Case Study

In this practice paper, we report the outcomes of a case study in a new Dutch hospital, where enterprise architects are working toward a 'lean' and 'simplified' EA model to align existing IT systems to new requirements. The objective of the case study was to examine if the developed EA model could support architects in selecting components of an existing IT infrastructure for re-use, with regard to radically new requirements. We have developed an EA model in close collaboration with enterprise architects. This study reflects on the use of this model in the hospital. The approach combines analysis of the content in the model, a study of documents in the organization, and communication with the architects. We signal that the existence of an integrated suite for an Electronic Health Record system largely determined how the model was used. Reflection disclosed that a lack of information on requirements and applications, as well as low adaptability of existing systems, negatively affected the flexibility of IT in the organization.

4.1. Introduction

Extant literature describes Enterprise Architecture (EA) as a promising approach to bridge the gap between Business and IT. In this paper we focus on aspects encountered in the practice of EA modeling. We explain the state-of-the-art concerning EA, at that time, by referring to a study from 2013 that gives an overview of the literature about EA in the period between 2003 and 2009 (Simon, Fischbach, & Schoder, 2013). In the literature, frameworks of EA are essential in research. TOGAF has been a de facto

This work was originally published as:

Tarenskeen, D., Hoppenbrouwers, S., & van de Wetering, R. (2018). Reflections on Using an Architecture Model for Matching Existing Applications to a Radical Business Requirements Change: a Case Study. In *IFIP Working Conference on The Practice of Enterprise Modeling*, 383-393. Springer, Cham.

guideline for practitioners. TOGAF is relevant for the case study, because it can be applied as a roadmap for the transformation of a Base Architecture (AS-IS) to a Target Architecture (TO-BE) as described in (TheOpenGroup, 2011). We refer to recent research, where new concerns for IT Flexibility on a strategical level have been described (Van de Wetering, Mikalef, & Helms, 2017; Van de Wetering, Mikalef, & Pateli, 2017; Van de Wetering, Versendaal, & Walraven, 2018).

This case study introduces a new specialized Dutch hospital (H1) in 2012, planning 600 beds. The hospital required the design of an EA with emphasis on the IT perspectives (Application Architecture, Information Architecture, Technical Architecture). As a guiding principle, the management decided that H1 should reuse the existing IT systems of a nearby academic hospital (H2). The development of the EA would be done by an Enterprise architect and an IT project manager (henceforth called "the architects"). H1 differs from H2 because it specializes in pediatric oncology. Also, H1 aspires to realize a vision in which child and family are positioned in the center of the care processes. Consequently, IT systems needed to be re-evaluated based on this basic assumption.

The case can be characterized as an exception in healthcare in the Netherlands, i.e., the possibility to start a new hospital (from scratch) is rare. However, the hospital builds on the existing infrastructure of a nearby hospital. Therefore, it will not really be built from scratch. The in-use IT systems are not typically legacy systems but are state-of-the-art systems currently (2018) in operation. The specific Dutch "Electronisch Patient Dossier" (Electronic Health Record System; EHR) solution is in use in more than half of all Dutch hospitals. The number of implementations of EHR-systems was growing in 2014, so EHR-systems can be expected to play a dominant role in the IT of other hospitals as well (Boonstra, Versluis, & Vos, 2014). The new hospital will be the central node in a network of hospitals. They will be working with "shared care hospitals" in the Netherlands.

We followed the design science research method (Hevner, March, Park, & Ram, 2004) in designing a model for the EA in close collaboration with the architects. In a first paper, we have presented the resulting model (Tarenskeen, Bakker, & Joosten, 2013). In the current paper, we reflect on the use of the model, and define the question for our research: *Does the developed model support architects in selecting the suitable applications for re-use*?

To answer this question, we analyzed the model and studied the decision-making process of the Board of Directors, informed by the internal advice reports and evaluations in the follow-up period. Hence, we reflect on the way the model is applied in practice.

4.2. Reflection on model in use

We evaluated two underlying purposes of the model. First, the model had to assist the architects with structuring all the information they collected concerning requirements and functionality of existing applications. Secondly, the model should give an overview of the relations between applications and requirements, in such a way that it helps architects decide on applications for re-use.

Two sub research questions were formulated (SRQ1 and SRQ2):

- SRQ1: Is the model suitable for structuring the information collected by the enterprise architects?
- SRQ2: How does the model assist the architects with selecting applications for re-use?

We reflected on the model during development with the architects (development phase) and afterwards by studying documentation (reflection phase).

4.2.1. Development phase

In the development phase the architects determined new requirements of the organization and the relations of the requirements to the existing IT infrastructure (in H2) by consulting 12 focus groups, the vendors of software, and the IT department in the organization. Examples of focus groups consulted are: Care Unit, Radiology and Radiotherapy, Education, Lab, Enterprise Principles, LATER (concerning health and complications after therapy). Architects provided the mapping of requirements to applications, based on the information they collected in the organization.

The EA model has been developed in iterations. In every iteration the researcher proposed a model and the enterprise architects evaluated the proposal and accepted or changed the model. We registered all drafts and discussed every adaptation with the architects. During development, insertion of data (instance pairs) was a regular activity. Examples of changes were: Adding, deleting or editing concept types names or relations. After the final model had been decided upon, all data considered important by the architects had been successfully inserted into the model. Inserting the data in the model was performed as a check for completeness of the model, and for suitability for structuring information provided by the model.

4.2.2. Reflection phase

After the development phase, an in-depth analysis was performed in the form of frequency distributions and content analysis of relations between requirements and applications. The objective of the analysis was to answer SRQ2, how the model could assist the architects with the task of selecting applications for re-use.

Also, we studied documentation, the reports of meetings of the steering group and architects, and reports of the 12 focus groups that formulated requirements. The results of the documentation study were discussed with the architects for answering SRQ2.

4.3. Description of the EA model

We selected the Ampersand business rules approach for modeling the EA because it can be applied to produce simplified models that have a mathematical foundation (Joosten, 2007; Michels, Joosten, Woude, & Joosten, 2011). The conceptual model for the EA is flexible and is defined in a separate script; the business rules can be declaratively defined separately in relation algebra, in the same script.

Ampersand is based on relation algebra and defines business information in a meta model. The meta model consists of descriptions of CONCEPTS, RELATIONS, PROCESSES and RULES (Tarenskeen, et al., 2013). A model is defined within a CONTEXT. All information is formulated in text, in the language of the stakeholders. There are no constraints on language for names, as long as the names are 'text'. Names have been used as identifiers. In this paper it is sufficient for the reader to interpret the Ampersand concepts and relations as similar to the entity and relationship names in an ER (EntityRelationship) model. The business rules in the model allow us to check the degree to which existing systems could fulfill new requirements. See section 4.3.3 Relations and Rules.

4.3.1. EA Model overview

Figure 4.1 presents the final model of the EA. The final model is the result of 12 iterations.

An explicit goal in designing the model was to include only the concepts needed by the architects for the specific (and somewhat unusual) architecture job at hand. We recognized: strategic goals (named Enterprise Principles by the architects), healthcare and business processes, and the services that are expected from the organization for supporting the processes. The services in the model were primarily application services (information technology services). The model has similarities with TOGAF architecture models. However, it is a simplified model because TOGAF models consist of separate extended models for Business Architecture, Information Architecture and Technology Architecture. In Fig. 4.1 the concept types and named relations are shown. The model includes both applications and requirements in one view.

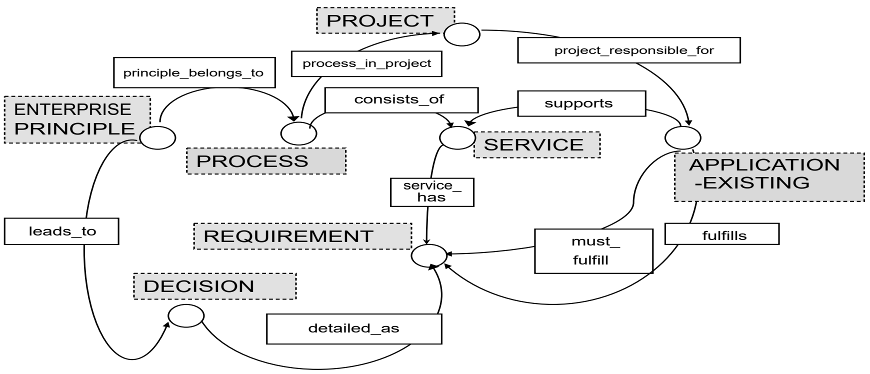


Fig. 4.1. Concepts and relations in the EA model (Version 12)

4.3.2. Details of Concepts.

In this section, all concepts in the Ampersand model are described.

ENTERPRISE_PRINCIPLE. An Enterprise principle is a principle that must be met according to the architects in the project (30 instances). Principles are derived from the healthcare vision of the hospital and can be compared to the strategic business goals of the hospital. Examples are “Intensive collaboration with medical staff and nursing care personnel and scientific research,” “Child and family will influence the care process wherever possible,” “Child and family will have the possibility to participate in education when in treatment.”

PROCESS. A Process is a primary process in the healthcare organization (11 instances). Examples are typical healthcare processes, such as Laboratory and Images, and Medication.

DECISION. A Decision is an Action or Goal that is based on an Enterprise Principle (43). Examples of Decisions: Visits must be planned, Reports of Anesthesia and Surgery must be made. It seemed that Decisions are sort of a dead end in the model because only relations to Enterprise principles exist.

SERVICE. A Service can be described as a composite service that delivers services for care processes in the healthcare organization (159 instances). Examples are the support for care processes, such as Diagnosis, but also Catering service or the Communication platform.

REQUIREMENT (OR QUESTION). Requirements are specific demands on an application or a combination of applications (132 instances). A large part of the Requirements have been marked Questions by the architects (75 of them). Some examples of Requirements (the remaining 57) are: “Formulate specific learning objective for every child”, “Functionality of patient portal is available through a website, a column in the building with electronic card, and via an app”, “The new and existing scheduling data from application X have to be transferred to another financial salary application”, “E-consultation”. Architects marked 75 requirements as “questions” that needed an answer before a requirement could be mapped to an application. That has resulted in two types of requirements, requirements and questions.

APPLICATION_EXISTING. This Ampersand concept concerns an application module or another component of the Information System (88 instances). An existing application is a software system that supports the care processes in the healthcare organization. A specific set of applications is the EHR, where all information about patients and treatment is stored and can be edited or retrieved. The EHR System accounted for a significant part of the applications that were defined. In the model, the EHR-system consisted of 39 modules, a ratio of 44 percent of all defined applications.

PROJECT. A project stands for a main EA project in the healthcare organization (1 instance). The architects have not reached the stage of formulating Projects.

4.3.3. Relations and Rules.

Relations are sets of tuples, instances of two concepts. A total of 2756 instance pairs have been formulated. Some relations were filled with only one, fake pair (1 pair).

When analyzing the Rules, we can conclude there are two kinds of rules. Rules for achieving the goals of selecting applications and rules for consistency of data in the model. For instance, all *must_fulfill* instance pairs have a similar *fulfills* pair. In Table 4.1. the relations[1] can be overviewed.

Table 4.1. Overview Relations and a number of pairs.

Relation	**Concept1**	**Concept2**	**Number of instance pairs**
fulfills	APPLICATION_EXISTING	REQUIREMENT	0
detailed-in	DECISION	REQUIREMENT	1
relates-to	PROCESS	PROJECT	1
responsible-for	PROJECT	APPLICATION_EXISTING	1
belongs-with	ENTERPRISE_PRINCIPLE	PROCESS	109
consists-of	PROCESS	SERVICE	225
supports	APPLICATION_EXISTING	SERVICE	249
leads-to	ENTERPRISE_PRINCIPLE	DECISION	251
must-fulfill	APPLICATION_EXISTING	REQUIREMENT	793
has	SERVICE	REQUIREMENT	1126

4.3.4. Analysis of requirements and applications

The researchers have performed an analysis after collected data had been inserted into the final model by architects. It was part of the Reflection phase. For this phase, we concentrated on the *must-fulfill*-relation since the *fulfills*-relation is empty.

To deepen our understanding and insights, we counted the number of requirements coupled to application modules and vice versa. We aimed to find the mapping relation, as a 1:1, 1:many, or many:many relation. The type of relation enabled us to see whether it is feasible to map requirements unambiguously to applications. For instance if every requirement only concerned one application and vice versa (1:1 relation) then the requirements were unambiguously related to one application and vice versa. First, we have checked if every requirement had been linked to only one application. This relationship demonstrates traceability of every requirement to an application. We could further investigate the degree in which the application did support the requirement. See Fig. 4.2.

When categorizing the *must-fulfill*-relation, we find a many:many relation between requirements and applications. Questions of architects also concern many applications each. Combining results of all counted pairs leads to the conclusion, that the requirements and applications are not structured in the same way, and that requirements cannot be mapped unambiguously to a specific application.

In related research, a framework has been developed that describes how to detect

[1] Relations are defined in Ampersand with two Concepts and a set of instance pairs.

Business-IT misalignment symptoms (Őri, 2016). When analyzing the misalignment of the TO-BE and AS-IS architecture with that framework, important indications concern organization and responsibilities, such as "S.01Undefined organizational mission, strategy and goals", "S.02 Undefined business process goals, business process owners", "S.03 Lack of relation between process goals and organizational goals". These symptoms of misalignment have not been found in the case study EA model. However, other symptoms regarding misalignment of Business process tasks and Applications arepresent in the model. For instance, "S.06 Application does not support at least one Business process" is signaled in the form of missing *fulfills*-relation in the EA model.

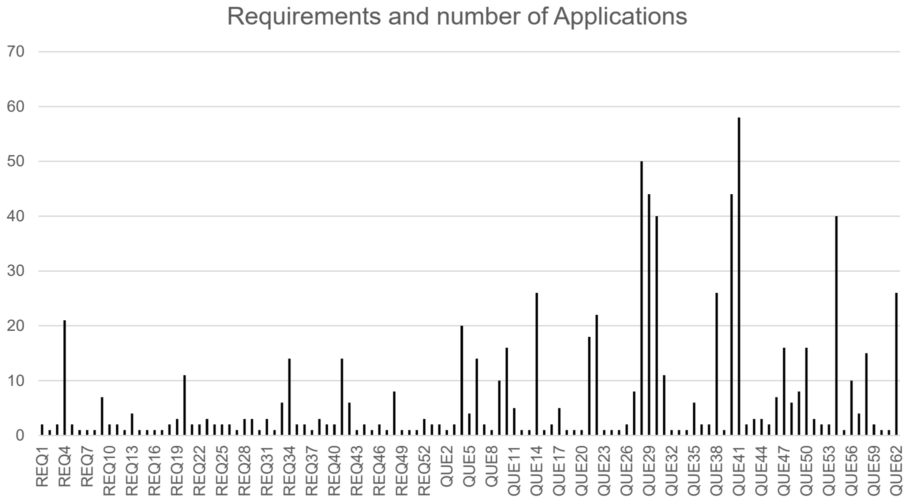


Fig. 4.2. Number of Applications with each requirement, final model

Also, the indication "S.07 Business process task supported by more than one application" is signaled. The *must-fulfill*-relation shows S.07 in the case study. We conclude, based on this analysis, that the simplified EA model signals misalignment of business requirements and applications.

4.4. Using the model

4.4.1. The role the model played in the decision process

In the follow-up after the model had been completed with data, we could observe the role the model played in decision making. It was decided by the Board of Directors and architects, that in the startup period (planned in 2014) the components of the IT systems were considered sufficient, unless medical professionals objected to specific components or applications. For their evaluation of the existing IT systems, the scope of the requirements had been restricted. Requirements related to many strategic goals were declared outside of evaluation scope, only working processes would be considered. The evaluation report stated that the IT infrastructure had sufficient capabilities to support the short time requirements for opening of the new hospital. Citation from Evaluation end report May 2013:

"The current, existing IT infrastructure is to a large extent sufficiently capable of supporting the working processes in the startup period. Some adaptations of application X are necessary for the adequate support of specific healthcare processes. These adaptations have to be realized in the period preceding startup period."[2]

The report describes in detail which adaptations in the IT infrastructure are a precondition for the new hospital. The bottom line is that standard working processes can be supported. Unfortunately, the model seems to have played only an indirect part in this decision, although the enterprise architects that made the model were on the selection committee.

However, healthcare processes that deviate from the work processes in other hospitals, such as prescriptions for medicine, preparation of medicine and registration of cytostatics for children were considered a risk.

4.4.2. Follow up evaluation by a working group of medical professionals

The steering group decided in April 2013 to start a new project for the period May-August 2013, to examine possibilities for developing an integrated (new) system for prescriptions of medicine, preparation of medicine and registration of cytostatics for children in the new hospital.

A working group investigated requirements and applications for the prescription process. From the documentation, the assumptions of the project participants became apparent. The working group set out with the following assignment [3]:

1. Describe the processes of working with protocols;
2. Define essential components in the current IT infrastructure that have to be included in the new IT systems;
3. Define criteria for scenarios;
4. Work out scenarios in detail to combine the functionality of different existing applications;
5. Evaluate the scenarios;
6. Advice the steering group of the most suitable combination of applications.

The working group reported a scenario that combined three existing applications, including applications that were not used in the existing IT architecture.

We can read in the report of the follow-up project[3] that a mismatch was found in the existing application landscape to support care processes for prescription of medicine. The medical professionals observed the same misalignment as was disclosed by the EA model.

[2] Internal Evaluation end report, May 2013

[3] Internal report Research of working with Protocols cytostatica v03, June 2013

Since the enterprise architects were part of the steering group, we assume the model played a role in the decision-making process, because we know that the model did provide an overview for the involved architects, though possibly an indirect one.

4.5. Conclusion and discussion

4.5.1. Research questions revisite

We answer the sub research questions in this paragraph. For SRQ1, we conclude that the model served the purpose of structuring information sufficiently, based on the test of inserting information in the model and discussions with architects. See 4.2.1.

The mapping of new requirements to existing application proved a challenge. The architects concluded that the mapping of requirements to applications could not be performed unambiguously, such that applications or modules could be selected for re-use. However, the model did provide the possibility for indicating that an application fulfilled the new requirements. See 4.3.2.

Therefore, for SRQ2, we conclude that the information that was collected by the architects was incomplete for selecting applications for re-use with the model.

4.5.2. Reflection and discussion

We observed that a large number of applications are part of an EHR-suite, or other integrated/interwoven application systems. Consequently, it was difficult to map requirements separately to each application module in the EA model.

We found numerous questions of architects about functionality, hence an indication that the documentation of functionality for specific hospitals is insufficiently accessible (if at all). As a further consequence, one cannot see how new requirements compare with the old ones.

Our study does confirm the findings in the study (Labusch, Koebele, Aier, & Winter, 2013), that a radical renovation of existing EA has to overcome the traditional structure of working and supporting IT. Our study adds new information by showing in some detail how the current IT architecture made up of integrated applications has obstructed innovation. It describes how lack of transparency and a modular structure that does not offer required flexibility, can obstruct innovation.

Our findings suggest that the core assumptions of architects: 1. insight in a fine-grained functionality in the applications, 2. flexibility of functionality for re-use and 3. transparency of the IT infrastructure, have been disproved.

It is too early to say that similar projects in other hospitals might lead to similar impasses because of similarity like IT systems. More research of EA in changing environments (of various kinds) is needed to extend the knowledge domain of EA to include adaptability, especially of the Application Architecture.

4.5.3. Recommendations & Future Research

We suggest that elaborate and costly efforts like the one in our case (business requirements, 12 focus groups, model), should lead to actual and explicit use of the model in the decision process (Return on Modelling Effort). Fortunately, it does seem to have been used indirectly, through the involvement of the architects in the decision process and steering group.

In a new study, we will perform a follow-up on this research by investigating how a separation of the data structure from the application layer can add to IT Flexibility. This research is likely to result in an expansion of the model of the EA model.

4.5.4. Limitations of validity of the research

This case study demonstrates clearly how architects struggle with many unknowns in the situation of modeling the EA for selecting applications for re-use. Since the case describes the situation of an organization startup, this could (partly) have caused some of the unknowns. However, the value of this case study lies in calling into question the core assumptions of architects and EA frameworks, such as the possibility of adapting existing IT systems and having complete access to information about IT systems and integrated application suites. If these core assumptions are not confirmed then IT Flexibility cannot be achieved by applying this EA model.

5 Conceptual Independence as an Architecture Pattern for Adaptable System

Conceptual independence is a novel approach for adaptable information systems. It extends the ideas of Model Driven Development. Conceptual independence eases adaptability by a separation of the Business domain terms from the functions that use the terms in the application. Although in practice many examples can be found in which parts of this approach are applied, a consistent application of this idea is not often encountered. The principle of Conceptual independence is sketched in this paper in a pattern. Since developing information systems often depends on the content of the Business domain, and the conceptual model of the domain needs frequent adaptation, systems need to be adaptable. In this paper we describe a complete decoupling of the software system from its conceptual model of the domain. We illustrate the relevance of this approach for the professional practice of software development and architecture. Furthermore, we describe related approaches for adaptable systems.

5.1. Introduction

Management of complex Business Information Systems, like the ones used in hospitals, is hard and involves much manpower. A Business Information System (BIS) is defined as:

"A group of interrelated components that work collectively to carry out input, processing, output, storage and control actions in order to convert data into information that can be used to support forecasting, planning, control, cooperation, decision making, and operational activities in an organization."(Hardcastle, 2011) (Bocij, Greasley, & Hickie, 2008).

In information and software architecture, much effort has been put to analyze and relieve this complexity, following familiar divide-and-conquer approaches such as defining layers and model-generated systems. In the last decade, management of BIS has grown in complexity, due to demands for connecting systems to other systems on the

This work was originally published as:

Tarenskeen, D. (2016, April). Conceptual Independence as an Architecture Pattern for Adaptable Systems. In *Proceedings of the 10th Travelling Conference on Pattern Languages of Programs,* 1-10. https://dl.acm.org/doi/10.1145/3022636.3022641

one hand, and connecting to apps and personal systems on the other hand. In practice the issue is far from solved yet (Tarenskeen, Bakker, & Joosten, 2013).

McGinnes introduced a new approach "Conceptual independence" (McGinnes & Kapros, 2015) for adaptive systems. The principle of Conceptual independence resembles Model Driven Development. It also separates the model of the code from the programming code. However it extends it, because the conceptual model of the domain is separated from the behavior in the domain model. A specific solution is offered for separating "behavior" from the conceptual model and separating both, from application logic. This solution is called Archetype. It is explained in the next example.The key points of Conceptual independence are:

(1) use a metamodel to isolate the terms and constraints in the business domain from the application logic
(2) enable change in functional requirements from the business without affecting other functional requirements, and
(3) make the coupling between functional requirements and software functions in programming code explicit.

As a result, complexity decreases, as changes in requirements lead to local changes in programming code. This approach follows the Axiomatic design principles of Suh (Suh, 1998) for designing adaptive technical systems.

In Axiomatic Design, the functional requirements are coupled one-to-one to parts of the technical system. It follows directly that in such a design in software, a change in requirements has only local effects in the programming code.
In section 5.3 we will describe the pattern of CONCEPTUAL INDEPENDENCE, but first in section 5.2, we will explain this pattern with examples. In section 5.4 the pattern is compared to standard patterns in software architecture and similar approaches in literature.

5.2. An example of conceptual independence

Before we describe CONCEPTUAL INDEPENDENCE, we will describe the terms that are related to the pattern and the most important relations.

The CONCEPTUAL INDEPENDENCE pattern is a pattern for software architecture. The pattern has the potential to have a great impact on the community of developers and therefore the understanding of this architecture pattern from a developer's point of view is crucial. Key terms we use are:

- Concept is similar to "Entity", examples of concepts are: Customer, Product, Order.
- Business domain: A part of a business described in natural language by stakeholders, relevant for the tasks that a person in business with support of a system will perform.
- Domain model: The representation of the business domain the way that software

developers represent it, usually in diagrams.
- Business logic: The constraints on concepts and relations in the business domain, for instance: multiplicity of relations, and number constraints.
- Business functions: Runtime functions that implement the business logic.
- General functions: Runtime functions that are unrelated to the business domain, such as infrastructural functions and middleware functions.
- Application logic: The structure in the system application, the components in the system and the way the components handle calls and use input and provide output. Application logic refers to logic in the system application at runtime.

In this paper specific terms are added:
- Conceptual model: All concepts, terms and Instantiations from the business domain and their relations to each other, in a formalized model, it does not include behavior
- Metamodel: An abstraction of conceptual models, the abstraction contains elements such as: "Concept", "Relation" and "Instance", in order to describe the conceptual model
- Soft conceptual model: An instantiation of a metamodel. The conceptual models are described as if they were data
- Archetype: A category of concepts in the conceptual model, that gives detailed directions for functions. For instance categorizing a concept as a Person can direct a programming function to present Person data on the screen, categorizing a concept as a an Address can direct a programming function to visually present a map with the address on screen.

The metamodel has a special place in CONCEPTUAL INDEPENDENCE, it defines the elements that are present in a conceptual model, such as Concepts and Relations. The conceptual model instantiates the elements of the metamodel. Therefore the metamodel must be complete, in such a way that the conceptual model (the business domain) can be meaningfully described. Ampersand, a method for modelling Business rules that is developed in the Netherlands at the Open University can illustrate how to construct an application in which the conceptual model is independent from Business logic. In the Ampersand language, which is based on Relation algebra, the conceptual model is based on a metamodel that gives freedom for modelers to construct a model of a business domain with few restrictions. Business rules can be specified for this conceptual model, and changed when needed. It is demonstrated below that only model elements as "Concept" and "Relation" in a metamodel suffice to describe a Domain in a meaningful way (G. Michels, S. Joosten, J. v. d. Woude, & S. Joosten, 2011b).

Table 5.1. Concepts and their instantiations

Concept	**Instantiation of Concept**
CustomerId	1244
OrderId	order9899

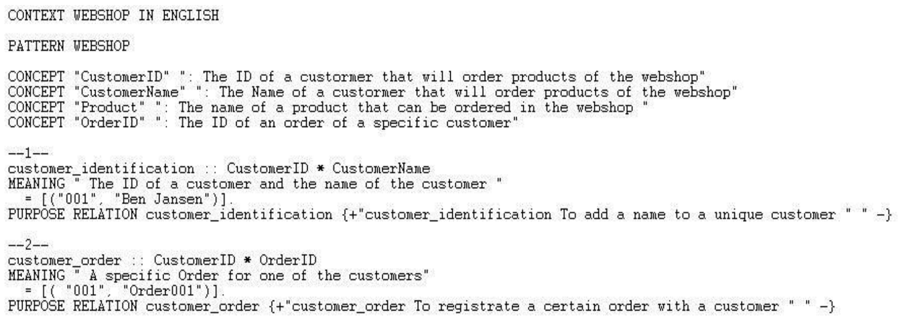
Table 5.2. Relation between 2 concepts

Relation	Concept1	Concept 2
customer_order	CustomerId	OrderId

Table 5.3. Instantiations of Concepts are shown in an instantiation of a Relation.

Relation	Instantiation of Concept1	Instantiation of Concept2
customer_order	1244	order9899

In Ampersand a conceptual model is described in a script that is a text file. The constraints on data and instantiations are declared separately in rules. Rules in Ampersand define which kind of multiplicities in Relations are allowed and which Relation combinations are allowed or forbidden. The syntax of the rules follows Relation algebra. In figure 5.1. a script of a part of a conceptual model is shown. The conceptual model is similar to the examples in the tables 5.1., 5.2. and 5.3., the data are different.



```
CONTEXT WEBSHOP IN ENGLISH

PATTERN WEBSHOP

CONCEPT "CustomerID" ": The ID of a custormer that will order products of the webshop"
CONCEPT "CustomerName" ": The Name of a custormer that will order products of the webshop"
CONCEPT "Product" ": The name of a product that can be ordered in the webshop "
CONCEPT "OrderID" ": The ID of an order of a specific customer"

--1--
customer_identification :: CustomerID * CustomerName
MEANING " The ID of a customer and the name of the customer "
  = [("001", "Ben Jansen")].
PURPOSE RELATION customer_identification {+"customer_identification To add a name to a unique customer " " -}

--2--
customer_order :: CustomerID * OrderID
MEANING " A specific Order for one of the customers"
  = [( "001", "Order001")].
PURPOSE RELATION customer_order {+"customer_order To registrate a certain order with a customer " " -}
```

Fig. 5.1. Part of an Ampersand script with Concepts and Relations

Ampersand uses Model Driven Development and extends Model Driven Development. Model Driven Development is applied in Ampersand for generation of applications. The Ampersand compiler can create database tables from the Ampersand scripts. The creation of database tables from text files or models is not new. It is similar in Model Driven Development (Völter, Stahl, Bettin, Haase, & Helsen, 2013). The database (RDBMS) with tables that contain the conceptual model is defined here as the Runtime conceptual model. Conceptual independence extends Model Driven Development when another Runtime that interprets the conceptual model is not used for creating database tables, but for executing other functionality. Runtimes based on the metamodel can be added and deleted without changing the conceptual model.

In Model Driven Development, the metamodel is hidden for remote calling procedures, once the database and other code is generated. From the outside, the generated application looks the same as a handmade application. This can be seen in use of object relation mapping frameworks (ORM), where code that is generated can be read by humans. Eventually the generated Model Driven Applications become similar to Object-oriented applications (OOA). The difference between Object-oriented Development (OOD) and Modeldriven Development (MDD) is, that OOD starts code programming with a stable conceptual model of the domain and MDD starts with the metamodel.

Here the Ampersand language is evaluated using the three criteria on page 54.The Ampersand language is Conceptual independent, because the conceptual model consists of data that is an instance of a metamodel of Concepts and Relations. Constraints are formulated in the form of rules, independent of application logic (page 54, 1). The functionality of rule checking can be added to the model as new functionality that does not change the conceptual model (page 54, 2). New rules can be formulated that are traceable to functional requirements (page 54, 3).

The system is extensible because other Runtimes can be added to the conceptual model that provide extra functionality, for instance for presentation of User interface screens. These other Runtimes need directions, for instance for User interfaces based on Archetypes.

In the Conceptual independence pattern Runtimes are dependent on the metamodel ins tead of on the Conceptual model. Therefore the conceptual model can be changed (independently) without having to change the functions that operate on it, or without recompilation. Rule checking behavior is decoupled from the terms in the conceptual model.

There are two types of functions, namely general system functions and functions that do depend on a particular business domain. First, general system functions, do not change the data per se, for example:

- Input
- Output
- Storage
- Read
- Write, without changing data

Second, business functions that do need the Domain concepts for working, such as:

- Update data based on Business logic
- Calculation of total sum of order
- Registration of customer.

BISs derive their value from the combination of general system functions and business functions.

To construct business functions one needs to provide these functions with directions, such as rules for rule checking. The functions not only need information about its operations, but also need input from the Business domain and necessary instantiations. That this is not exceptional is seen in REST web services, where documents, in for instance JSON, deliver all information about the data that the function needs. The documents describe the Concepts and Relations. They do not have to be hard coded in the REST service.

With Conceptual independence the conceptual model is data, the metamodel is explicit and functions operate on the metamodel, therefore functionality can cope with *any conceptual model that is based on a specific metamodel.* This is the contribution of Conceptual independence.
See pseudocode.

An example of pseudocode presents code for rule checking functionality, which needs a rule in a specific syntax and data from the conceptual model.

Algorithm description: This rule checking functionality of the Rule Engine, knows the syntax of the rules, can import enough information from the conceptual model to be able to operate. Without needing to know in hard coded parts, the contents of the model.
Parameter1 = Rule specification in known syntax with Domain specific information
Parameter2 = Part of the Domain in format of metamodel or in standard JSON

```
check_rule (Parameter1, Parameter2) returns Violations
begin
   do
               rule = parse(Parameter1)
               internal_representation_of_domain = parse(Parameter2)
               relevant_concept_instantiations =
                  select (internal_representation_of_domain)
               violations =
                  execute_rule(rule, relevant_concept_instantiations)
   return      violations in document format
end
```

Another example that demonstrates the flexibility and versatility of Conceptual independence is seen in a spreadsheet application where calculations can be performed for every Business domain, with the same runtime spread sheet application. The metamodel consists of the matrix for rows and columns. The conceptual model can be inserted as data in the matrix. In other cells in the matrix rules and formulas can be inserted, that have a reference to specific cells in the matrix. The spreadsheets does fulfill two of the three requirements for Conceptual independence:

(1) use a metamodel to isolate the terms and constraints in the business domain from the application logic: values in the spread sheet are separated from execution logic in the spreadsheet application
(2) enable change in functional requirements from the business without affecting other functional requirements: changes in functionality can be formulated in formulas and rules
(3) make the coupling between functional requirements and software functions in programming code explicit: this is not supported by spreadsheets.

It is possible to configure specific formulas in the spreadsheet without changing the spreadsheet application. Without recompilation.

5.3. A pattern for conceptual independence

In this section we will describe CONCEPTUAL INDEPENDENCE in Alexandrian style (Alexander, Ishikawa, & Silverstein, 1977).

Conceptual independence*

We will use the following inbound pattern: An INTERPRETER. A runtime component for generating program code or scripts is added. In this way we can separate Business Domain and application logic. INTERPRETER is comparable to a known pattern as described by Buschmann (Buschmann, Henney, & Schmidt, 2007b) (Page 442). According to Buschmann the INTERPRETER reads "...data or scripts, has grammar knowledge and executes the right services on the components". This is exactly what happens here. The INTERPRETER directs the generator to generate code or scripts. A generator for building the business application could be compared with ABSTRACT FACTORY (Page 525) and BUILDER (Page 527). The generator in CONCEPTUAL INDEPENDENCE is used for generation of programming code and does not lead to a Runtime changes in the generator. The generator acts similar to a program for model transformation that transforms a model to text. The conceptual model of the domain does not contain behavior.
When you need to develop a BIS in which functionality will frequently change

... you should design your system such that conceptual model of the domain and business logic are easy to locate and change.

♦ ♦ ♦

Software is often structured top down with the conceptual model of the domain hard coded in the software program. BISs usually are built in this way with a conceptual model of the domain as a starting point. This makes adaptation of the program difficult. This is the problem CONCEPTUAL INDEPENDENCE addresses. The coupling of the conceptual model of the domain to application logic, the top down approach, makes it difficult to apply changes after the initial program is realized. References to the conceptual model of the domain can be found deep in the methods of the program code, and are often spread across the program.

The standard software development process needs a stable conceptual model of the domain before software coding can start. Behavior in the domain is represented by business logic. The business IT analyst is responsible for a consistent and stable view on this domain. A technical architect is responsible for drafting an architecture, supporting and structuring code development (McGinnes & Kapros, 2015; Singh, Bhattacherjee, & Bhattacharjee, 2012; Zhang et al., 2014). It is a de facto standard to implement business information in program code at various spots. It takes implementation knowledge to access that information or change it (Fowler, 2002; Garlan, Allen, & Ockerbloom, 2009; Gotel & Finkelstein, 1994).

Because standard software development needs to know beforehand how the conceptual model of the domain will change and this is often not the case, we need another solution.

Therefore…

Provide a system design in which the conceptual model of the domain and business logic are separated from the low level code and from application logic, in such a way that the conceptual model of the domain and business logic can be changed without disturbing application logic. And provide room for adding, editing or deleting application logic without having to change the conceptual model of the domain.

In Fig. 5.2. an overview of the essential parts of CONCEPTUAL INDEPENDENCE, without functionality, is shown. We will explain the different parts. And then add behavior for the conceptual model.

5.3.1. Base pattern Conceptual independence and MDD

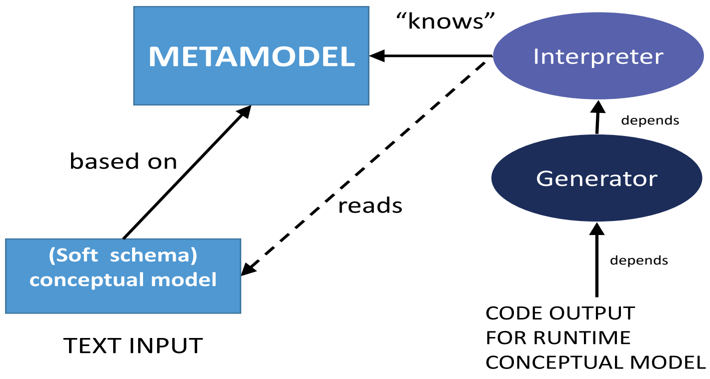


FIG. 5.2. Base Pattern for CONCEPTUAL INDEPENDENCE and Model Driven Development

In the model in Fig. 5.2. above, the arrows show dependencies. At the left the conceptual model is represented in Text files described with terms defined in the metamodel, for instance: "Concepts" and "Relations". With these two meta concepts most common conceptual model of the domains can be described. If one wants to add Attributes, Attribute can be added as element to the metamodel. The Soft conceptual model is called "soft", because it can be changed independently from the INTERPRETER and generator runtime that reads it.
This Base pattern is similar to the way Model Driven Development works, and is similar to the approach of Yoder, see Section 5.4.

The base pattern for CONCEPTUAL INDEPENDENCE consists of:
- A Metamodel (format) with the structure and elements in the Soft conceptual model
- The Soft conceptual model
- An (Domain independent) Interpreter
- A (Domain independent) Generator or a Function component
- Separate Code or scripts for the Runtime
- Or Output from the Function Component
- An Archetype and its specification

The Base Pattern for CONCEPTUAL INDEPENDENCE, must be extended with a mechanism for creating functionality for the Domain. In the principle of Conceptual independence McGinnes augments Domain Concepts with an Archetype, this in fact is sort of a Functionality outside of the conceptual model of the domain that links the Domain Concepts to functionality. An Archetype can for instance be "Rule", which opens the possibility to define specific rules for every conceptual model, with one "rule checking component". In Fig. 5.3. Archetypes are added to the base pattern of CONCEPTUAL INDEPENDENCE.

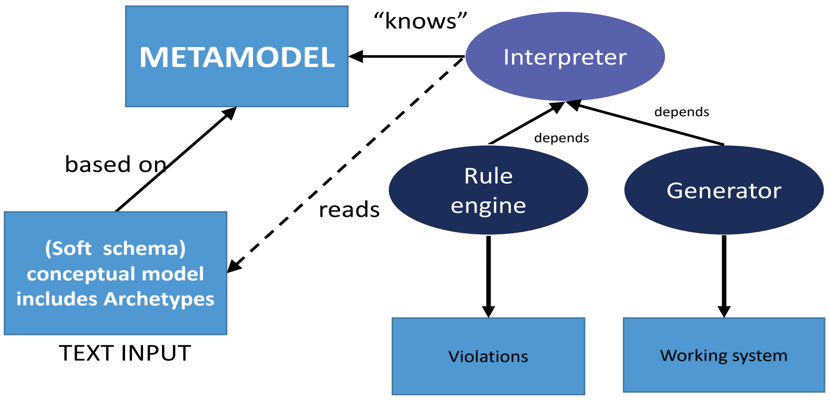


Fig. 5.3. Conceptual model of the domain with added Archetype Rule

The INTERPRETER reads and interprets the information of the Soft conceptual model with Archetypes included, and invokes the Rule Engine. The Rule Engine has knowledge of coupling the right "formulas" to the domain concepts and outputs violations. We can thus create functions that are independent from each other. We recognize the approach of Axiomatic design in the independence of functional requirements (Suh, 1998). Suh advocates in his ideas for Axiomatic Design that adaptability of industrial products is achieved when all functions in a product are implemented with separate "design features". In our example, each of the functions can operate on the data, or use system functions on the data without causing side effects, that could affect other functions.

The benefits of this pattern, are that the (soft) conceptual models and the (soft) Business logic can change without affecting application functions. It opens possibilities to design software systems in which every high level function is independent from other high level functions. Other benefits are that Business functions and General system functions can be changed or added without changing the conceptual model.

In Fig. 5.4. the resulting application is represented by the symbols for Runtime components: Runtime Archetype functions, Runtime Conceptual model and General system functions. In this Runtime model we presume that Archetype functions and General system functions use an API of the Runtime conceptual model.

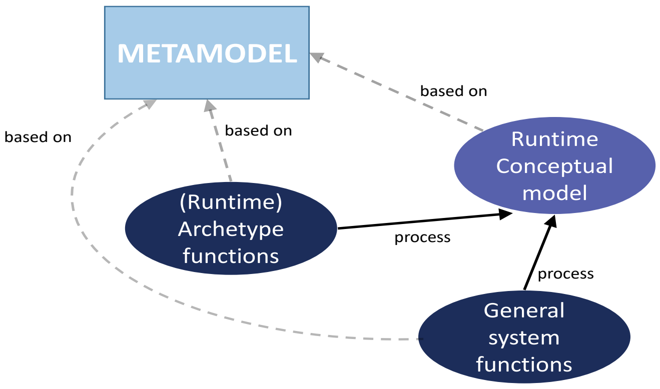


Fig. 5.4 Representation of the Runtime Components of the business application

5.4. Related approaches

Some approaches have been developed for improving adaptability of BISs, especially where the conceptual model is concerned. We think of the Adaptive Object Model, Model Driven Development (Völter, et al., 2013), software factories and product lines for software (Czarnecki, 2005), metadata-based frameworks such as Hibernate and Rule Based Systems. We will discuss the Adaptive Object Model and the Ampersand Rule Based System in this paper. In section 5.4.3 patterns in Buschmann (Buschmann, et al., 2007b) that have characteristics similar to CONCEPTUAL INDEPENDENCE are discussed.

5.4.1. Patterns of Adaptive object model

In the idea of "Adaptive object model" (AOM) Yoder (Yoder, Balaguer, & Johnson, 2001) defines that an "AOM is an architectural pattern based upon a dynamic meta-modelling technique where the object model of the system is explicitly defined as data to be interpreted at runtime". This is further explained by Ferreira in an extensive description of types of meta-modelling and system development (Ferreira, Correia, Aguiar, & Faria, 2010). In Fig. 5.5., Ferreira describes the TYPE-OBJECT Pattern. This points to the Basic Pattern for Adaptive object Model.

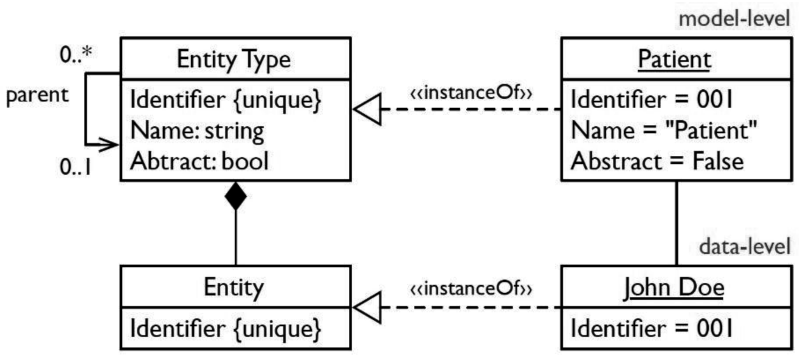


Fig. 5.5. TYPE-OBJECT Pattern from Adaptive ObjectModel

Entity Type and Entity in this example are model elements of the metamodel.
Every Entity Type instance can be added to the system through data input, and the same is possible for instances of the Entities. A pattern for storing and retrieving meta-data is found in the paper of Yoder (Yoder, et al., 2001). The conceptual model and the model of the application are defined in XML/XMI. The XML text files are interpreted. The application provides all functionality.

There are two main differences between this approach and the approach of Conceptual independence. One is, that in AOM there is no distinction between declaring the conceptual model of the domain, and declaring the functionality whereas Conceptual independence distinguishes the conceptual model of the domain and the Archetypes. The other difference is that the scripts in AOM are interpreted and are part of an integrated Object oriented system, with forward engineering, whereas in Conceptual independence the parts can distinguished and extended from outside.

5.4.2. Patterns in Rule Based Systems

In Rule Based Systems according to the Ampersand approach (G. Michels, S. Joosten, J. Woude, & S. Joosten, 2011a; Michels, et al., 2011b), the same Base Pattern of CONCEPTUAL INDEPENDENCE is found. The soft conceptual model is described in scripts (texts), separately from the functionality. Functionality is formulated in rules in relation algebra. Rules play the role of Archetypes such that they define functionality of the conceptual model of the domain.

The functionality of the application is rule checking, a Rule Engine is responsible for checking rules represented in Relation algebra. The process of rule checking results in violations. A conceptual independent Rule Engine is realizable, as is demonstrated with the approach of Ampersand (Michels, et al., 2011b). Ampersand however, also uses MDD to create a second Rule Engine, which operates on the Runtime conceptual model implemented in a database. The program code of the second Rule Engine does depend on the conceptual model. It must be re-generated when rule definitions change.

5.4.3. Buschmann architectural patterns

Existing patterns such as DOMAIN OBJECT and LAYERS (Buschmann, et al., 2007b) (page 208, 185) can fail to deliver ease of maintenance, because implementations of these patterns do not completely decouple concepts of the business domain from the program code. They do not use a metamodel to base the functions on.
A separation of meta level and Object level is described by Buschmann in REFLECTION (197). This pattern is based on DOMAIN OBJECT and has the same problem as mentioned above. Content of the objects must be known at programming time. In CONCEPTUAL INDEPENDENCE, however, the concepts do not need to be known at programming time, they can change anytime when the conceptual model is (re-)designed.

5.5. Conclusion

We have presented the Pattern of CONCEPTUAL INDEPENDENCE and its argumentation and consequences. We need to build experience with practical applications of this pattern. The first experiments of McGinnes demonstrate the possibility of realizing systems that conform to the pattern. We can build Adaptive systems that are not "…tied to a particular conceptual model." (McGinnes & Kapros, 2015).

5.6. Future research

In the future a conceptual model can take the form of an Ontology. Standards as RDF and OWL formats can be applied. Functionality for the Domain is accessible through the Archetypes. Concepts in the Domain can belong to one or more Archetypes and "inherit" functionality in this way. This design for system development is described in Ontology-based Adaptive information systems. This extension of CONCEPTUAL INDEPENDENCE is described by Bhérer (Bhérer, Vouligny, Gaha, Redouane, & Desrosiers, 2015).
In the future the possibility of applying conceptual independence in a system-of-systems architecture must be studied. For instance, Conceptual independent system components can be programmed or generated that use REST-full communication protocols. The REST-full communication can be applied for the Runtime conceptual model and for the general or business functions. This opens up possibilities for distributed systems, with loose, independently runnable components that provide added functionality to a soft conceptual model.

Acknowledgements

The paper is written in the context of the VikingPLOP Writer's workshop 2016 and I would like to thank the participants of the workshop and Pietu Pohjalainen, Department of Computer Science, University of Helsinki, Finland, for their extensive reviews. The ideas in this paper and the illustrations of the paper that are based on Ampersand have been discussed with Stef Joosten, Open University, Heerlen, The Netherlands. Also a specific thanks for René Bakker of HAN University of Applied Sciences. Hereby I express my gratitude for their contribution to the paper.

6 Applying Axiomatic Design and Conceptual Independence in the Domain of IT Systems

In this paper we explain why Axiomatic design has not been applied in large system of systems information technology architectures in health care organizations in the Netherlands. We have found in extensive case studies of IT systems that the Independence axiom could not be found in the existing Information systems. This causes great concern for the adaptability of systems. Although best practices in system engineering advice decoupling of system functionality, findings show that the Independence Axiom has not been applied to functional requirements. A number of difficulties was exposed to the researcher and IT architects, when they tried to identify and to demarcate functional requirements in existing systems. In IT systems a distinction is made between business and system functions and software engineers emphasize decoupling of system functions. That has resulted in a strong coupling of business information and technical application logic. This means Conceptual dependencies are often at the heart of the difficulties. Decoupling these dependencies in applications seems a candidate for ordering requirements in a way that keeps the functional requirements independent from each other. Conceptual independence signifies a complete decoupling of the concepts and relations of the Business domain from the technical functionality. The paper describes the difficulties when applying the Independence Axiom in the Information systems domain and argues that a combination of Conceptual independence and Axiomatic design is achievable. The authors adapt the Design matrix with Conceptual independence. We conclude that applying Conceptual independence is crucial when constructing IT systems based on Axiomatic design.

This work was originally published as:

Tarenskeen, D., & Bakker, R. (2017). Applying Axiomatic design and Conceptual independence in the domain of IT systems. In *MATEC Web of Conferences,* 127(01006). EDP Sciences.

6.1. Introduction

In this paper we will argue that Axiomatic design cannot be applied in IT systems consisting of many subsystems, because of Conceptual dependencies. In a paper of Suh in 1998, he argues that Axiomatic design can be applied in system design for different kinds of systems, including systems for Information technology (Suh, 1998). The authors found in three case studies that IT systems were hard to change. They examined the IT systems and detected that Axiomatic design had not been applied. In the next section we will outline the three case studies in health care. In section 6.3. we will explain that Axiomatic design was applicable in the cases according to Suh (Suh & Do, 2000), but could not be found in system architecture diagrams. We even encountered difficulties when trying to restructure the systems according to the Independence axiom.

Next in section 6.4. we will extend on the aspects that hindered direct application of Axiomatic design. The aspects show an interdependence between technical functionality and business terms. We call this interdependence Conceptual dependence conform McGinnes' definition (McGinnes & Kapros, 2015). The principle of Conceptual independence is explained in section 6.5. We propose a model for system architecture of IT systems with independent components in section 6.5.

In section 6.6. Conceptual independence is positioned in the Design matrix of Axiomatic design. We conclude with stressing the importance of applying Conceptual independence when constructing IT systems based on Axiomatic design.

6.2. Description of case studies

The authors have studied three cases in health care in the Netherlands in the period 2012-2015 where existing systems needed to adapt to new health care requirements and processes. In the organizations a complex IT infrastructure functioned sufficiently and reliably for the current care organization. The board of directors in all three cases wanted to re-evaluate the IT systems in the light of new requirements. In all case studies we developed models of the IT architectures in collaboration with the IT architects in case 1, 2 and 3 or with the IT department in case 2. When compared to de facto IT architecture methodologies for change, the end goal of the IT architects was a Roadmap for adapting the existing IT system architecture to an IT architecture suitable for new requirements. This way of working resembles the TOGAF method in IT architecture practice (TheOpenGroup, 2011).

6.2.1. Cases in health care

Two case studies were performed in hospitals, one hospital was still in the design phase and was planned to be a central node (with a physical building in Utrecht) in a network of care centers. The latter were departments in already existing hospitals. The other hospital was a regional medium sized hospital. The remaining case study was performed in a Home care institute that also provided facilities for intern patients and the elderly (Tarenskeen, Bakker, & Joosten, 2013, 2015).

The first case study took place from November 2012 until April 2013, the second from March 2013 to November 2014, the last from February 2015 until June 2015.

Examples of new requirements were:

- In all processes patients and their family are primary,
- Care processes and information must be organized around the patient and its family and not around medical specializations,
- All Information that is involved in the invoicing process must be integrated within the invoicing process as a whole,
- Achieve an increase in efficiency in planning of diagnosis and treatment with IT,
- Supporting IT for workflows of persons involved in planning of the ordering process of diagnosis and treatment.

6.2.2. Case study 1 National hospital for child oncology

The reason for developing a new system architecture was the establishment of a new national hospital for child oncology, that would integrate laboratory functions and treatment, integrate knowledge from research in this area and treatment and coordinate care centers across regions in the Netherlands. Information technology that existed in an academic hospital nearby would be reused where possible (UMC Utrecht, 2012). The IT architecture model, that was eventually selected by the IT architects can be seen as a simplified version of an enterprise architecture in which strategic goals were detailed in requirements for applications. In the Requirements analysis stage all strategic goals were specified into Services and those were specified in requirements for every one of the 90 applications.

6.2.3. Case study 2 Home care organization

The organization in this case functioned mainly as a care organization and provided home care and care for the elderly. It had 5000 employees and about 3000 volunteers at the time of study (Stichting Carint Reggeland Groep, 2014). The IT architect and IT expert started with formulating strategic goals, because the organization was in the process of changing its health care vision and strategy. The ultimate goal however, was to judge if the current system was up to the required changes and could be maintained in its current form. The Board of directors was planning to update and replace parts of the system that were not functioning according to state of the art requirements.

6.2.4. Case study 3 Medium sized regional Hospital

In the third case study a regional hospital in the Netherlands is being studied (Interconfessionele Stichting Gezondheidszorg Rivierenland Culemborg, 2013). The hospital is in the last stage of changing its paper-based operations to digital operations in the health care processes. The goal of the IT architects is to deliver a consistent vision of the changes in health care services as foreseen and the specified supporting IT functionality.

In the case studies the methods for changing existing systems were examined and notated. Practice showed that systems were difficult to change. We examined if in the existing systems the principles of Axiomatic design were applied.

6.3. Application of the Independence Axiom

Since in all three cases IT architects needed to adapt systems to new requirements, the lines of demarcation in the Information system architecture are crucial knowledge.

Our scope of Information system architecture is not confined to one Information system only. We have found that the IT systems in health care include Electronic Health records, in the Netherlands called Electronic Patient Dossier, systems for financial administration, systems for management, systems for special equipment such as MRI scan, systems for Enterprise resource management, systems for procurement.

6.3.1. Functional Requirements

In Information technology a distinction can be made between system functions and business functions. System functions are necessary in all organizations that need information for the working process such as storage of data, searching in collections of records, exchanging data between servers, exchanging data between servers and applications.

Business functions are functions that are useful in a specific organizations or industry, such as accessing a Patient Dossier, accessing and analyzing medical diagnosis, registration of a new patient, planning medical treatment in health care. In these functions specific health care terms are included in the system functions. Often the system functions can only be valuable for health care if information in information systems can be accessed with health care terms.

In papers of Suh regarding system design, he emphasizes the zigzagging over different domains when designing the definitive system (Suh, 2001). In this paragraph I will explain how zigzagging over domains is applied in IT Requirements analysis. The Customer domain in IT consists of needs of the board of directors, but also of the medical specialists and other care professionals, such as nurses, and the information needs of patients. The iterations of Customer domain to the Functional domain in the ideal case will lead to independent Functional requirements. Then it will be possible to link the FRs to the Physical domain. The preconditions for zigzagging from the Customer Domain to the Functional domain in Information system design are fulfilled. In case 1 the Requirements analysis was performed with 11 working groups of stakeholders. It resulted in 133 requirements with 793 Relations to Applications.

6.3.2. Decoupling Design parameters in IT

We have found that the Independence Axiom could have been applied in Information systems design in the same way as Axiomatic design prescribes. This is possible, because in realizing and programming of IT systems decoupling modules and functions by way of interfaces between different components in a system is good practice in software engineering. Patterns of decoupling system components are seen as qualitative high standards of software and are described in standard IT publications, such as in Larman (Larman, 2005), Buschman (Buschmann, Henney, & Schmidt, 2007b) and Bass (Bass, Clements, & Kazman, 2012). Therefore we have concluded that Axiomatic design can be applied for defining independent or decoupled elements as Design parameters in IT systems in health care.

6.3.3. Conclusion of applicability of Axiomatic design

If we take the statements of IT engineering at face value, we can conclude that Axiomatic design can be achieved in designing and realizing IT systems. In the next section we will explain that demarcation lines in IT systems do not follow functional requirements.

6.4. Demarcation lines in IT systems in case studies

6.4.1. Case study 1 and 3: National hospital for child oncology and Regional hospital

Because we wanted to find the FRs of the existing systems in IT system architecture, first we have examined the most important FRs.

In case 1 in the hospital new requirements were formed because the vision of care as care for a patient and its family led to a changed vision of IT support. The IT support should adapt to organizing the systems around the patient and its family, instead of organizing the applications around medical specializations, its administration and registration needs. In case 3 strategic goals were specified in changing certain work flows for care processes. The new work flows should include communication and exchange of data with medical professionals from outside the hospital.

We have sought demarcation lines in IT systems for these FRs, and distinguished independent Runtime components in IT systems. The demarcation lines in existing systems in the hospitals were based on “applications” provided by suppliers for Information technology for health care.

We conform to the definition of application in this case study used by the IT architects in the study. An application can be defined in two ways:

1. A computer program that can be installed and run independently of other computer programs, and that delivers functionality defined by the supplier. The application can sometimes be coupled to medical equipment or can use open standards for communication. Features for network connections and communication over the network in proprietary standards and protocols are usually present.

2. A loosely coupled component in a computer program can also be an independent component, e.g. a module or functionality that can be turned off or on, depending on the possibilities that the supplier offers.

We have found in the cases that most of the functional requirements have been implemented in the applications in a way that is historically grown, and based on the knowledge that the suppliers had of the business processes of their customers, the health care organizations in the past.

Remarkably in both Hospitals was that a central position in the IT systems was reserved for the Electronic Patient Dossier. However the data belonging to the EPD was distributed among multiple sub applications that each have access to a part of the data of patients.

In both hospitals the IT system architecture consisted of applications that did not follow the distinctions between FRs.

6.4.2. Case study 2: Home care and cure

In the second case study we were interested in the divisions that IT technical experts were making when viewing the IT system architecture. We researched methods of change in IT system architecture in collaboration with technical IT architects.

An example of part of the IT system can be seen in Figure 6.1. In this figure the components in the system, do consist of applications (A, E and F), however these and the other systems are executing system technical functions, such as: storage of employee data and scheduling data, and exchange of data between one component and the other.

We can point out the following elements from part of the IT system architecture:

- A: The mobile application for input of employee data,
- A-B: A connection from the mobile device to the first proprietary database,
- B: The first database that communicates with the mobile device and stores the data in the IT systems,
- B-D: A connection from the first proprietary database to a second more general database,
- C: System that couples data to dashboards,
- D: The second database that collects all data from the first data storage database,
- D-E: Data output to combine data from data storage database to general information,
- E: The database that collects all general information, about history of illnesses, diseases, base tables. Also administration of invoices and finances. Indications for care in relation to insurances,

- E-F: Output of data from E to database F,
- F: The database that stores financial data for yearly reports.

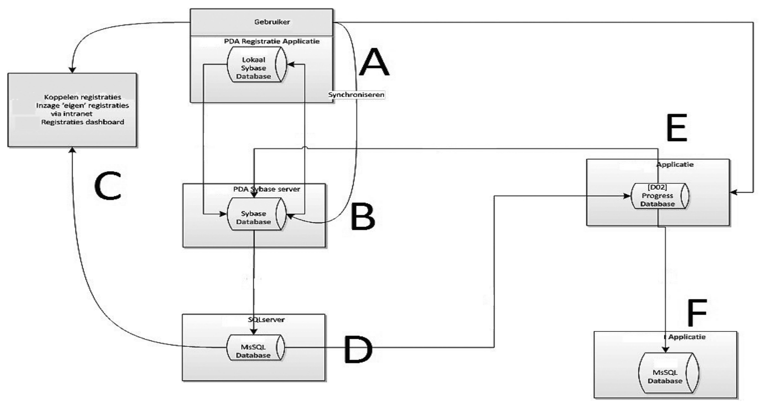


Fig. 6.1. Components in Home care institute.

The components are partly delivered by suppliers, partly these are custom made components. We add a third definition for an independent element in IT system architecture:

3. A computer program that can be installed and run independently of other computer programs, and that delivers more general system functionality. Examples are: data bases, input systems, file systems, hardware drivers, software components that enable network communication.

In this case the diagram focuses on system functions (Tarenskeen, 2016). Applications are only named applications in this context when interaction with users is performed.

6.4.3. IT Actions for adapting systems

We will list the different IT actions in Table 6.1. that can change business and system functionality: Add a component, delete a component, change a component. However there is a fourth way of changing the IT systems, that is well known in software engineering as configuring systems with configuration files or in registry systems on the machine. Configuration are based on a structure that can be simple or more complex, they can even take the form of a conceptual model, with concepts that are structured and related. With changes in the content of the configurations (file) the behaviour of the Runtime component is changed without changing the component itself. Runtimes like these are called "computer language interpreters" or interpreters for scripting languages.

In all cases in Table 6.1., described above the changes must be foreseen. A component can only be added to a system when collaboration with other components is built in, in all system components that are concerned. A component can only be deleted when it does not contain functionality expected or needed by other components (dependencies). A component can only be changed, when changes do not affect communication with other components, or influence configurations of other components.

Table 6.1. Changing system by changing components and connections.

Behavior change	**Structural change**
Add component	Add communication line
Delete component	Delete communication line
Change component	Change communication line
Change configurations	Change structure of communication

A configuration of a component or a model based configuration file can only be changed in predefined ways, as the component "expects" configurations, and cannot cause unexpected or new behavior (Garlan, Allen, & Ockerbloom, 1995, 2009), unless something goes wrong.

When the behaviour change concerns more than one component, communication between components must be changed. Connections can be added or deleted, thereby influencing the flow of information through the system. With the Independence axiom adding, replacing and deleting components or connections based on changing FRs will be feasible. We propose in the next chapters that also the option of "changing configurations" must be taken into account when designing Information systems.

6.4.4. Difficulties encountered when changing systems

In the cases that were studied we have found that the IT architects in comparison to experts from IT departments used different conceptual models for looking at the IT systems. IT (Enterprise) architects started with the strategic goals and vision on health care in mind and proceeded from there with defining business processes, such as in health care: Diagnose, Provide medical results, Propose treatment, Care plan, Medicine plan, Monitor and measure. These processes will be supported by IT services, and Functional requirements can be formulated as Functional requirements for these IT services (Tarenskeen, et al., 2013). The technical IT architects however, were looking at systems of system components and were interested in system failure and system functionality.

Despite the different views on the system, all cases showed that interdependency of conceptual data and system functionality obstructed required changes.

6.4.4.1. Patient data distributed and structure unknown

In the first case study, the changes required reorganization of applications, because the data for patients and their family was needed in a central place. IT architects found that data was distributed over different applications, and that the overall structure of patient data did not reside in one place, so to have access to the patient data structure from each application as needed. The data was structured in such a way as to accommodate the work flow and processes of medical specialists. Functional requirements were coupled to applications in a many to many relational structure. Therefore we conclude that the Independence axiom was not applied.

6.4.4.2. General system components adaptable

In the second case study the system consisted of system components and general system functions such as: storage, exchange of data and software for input devices. All components were dedicated to one system function, in which "business data and business information" were encapsulated.

The Functional requirements were not easy to locate in a system were information structures were hidden from view. Therefore we conclude that the Independence axiom had not been applied.

6.4.4.3. Work flow data distributed and work flow steps in applications are fixed

In the third case study in the Regional Hospital the IT architects sought to reorder the processes in which patients visited different departments in hospitals for medical examinations, medical results and diagnosis. The planning of these visits to different locations, the order in which the visits were planned was subject of study.

It was found, that although the application platform had been recommended as flexible, especially where workflows were concerned, the reorganization of workflows was not easy. The main cause of this, was the fact that documentation of the workflow formalization was difficult to understand. IT architects wanted to try out alternative workflows, but there were no tools supporting this. Functional requirements were not traceable in the previous workflows and we concluded that the Independence Axiom had not been applied. A clear distinction between Business concepts and technical implementation of workflows had not been made.

The problem of interdependencies of Conceptual models and technical logic in applications has been remarked and described as Conceptual dependence by McGinnes, we explain this in section 6.5.

6.5. Conceptual independence

6.5.1. Conceptual dependencies

McGinnes describes how conceptual dependence can limit the changes one can make in independent elements in a system (McGinnes & Kapros, 2015). He describes Conceptual dependence as interdependency between the terms in the Business domain and the technical implementation of the application. In the previous paragraph we have shown that difficulties in adaptabilities in systems are the result of conceptual dependencies. We argue that a decomposition of functionality without separating the Business data from the system is insufficient.

Conceptual independence is based on separating the conceptual model from the technical application logic. A Conceptual model includes: all concepts, terms and instantiations from the business domain and their relations to each other, in a formalized model, it does not include behavior.

Conceptual independence is based on a meta model, in which the Conceptual model is filled in as data. For instance, the meta model consists of: Concepts, Relations, Attributes and Instances of concepts.

The characteristic of Conceptual independence is that the Conceptual model can change, without having to change the technical system that runs with the Concepts. Because the Conceptual model is an instance of a meta model, it can be treated as data, we call it a soft model (McGinnes & Kapros, 2015).

In a previous paper describing the pattern of Conceptual independence, we propose a complete division between the conceptual model where all business terms are collected in an Entity relationship-model (ER Model), as is customary in database design (Tarenskeen, 2016). We then make a distinction in functions that are totally determined by the business and system functions that are general functions that can be applied also in other organizations. The latter need to be configured to this context/business. We propose the configuration of general functions instead of building functions anew for each organization, see model in Chapter 6.5.3. In 6.5.2 we give a recognizable example of Conceptual independence.

6.5.2. A Relational database management system as example of Conceptual independence

A well-known example of a Conceptual independent "application" can be found in a Relational Database Management System (RDBMS). This system can be deployed in any organization and filled with tables and table names specific to this organization. It can also store specific Business requirements in stored procedures and triggers that are fully determined by the business as concerned. These are functions that are programmed in a specific Data base language, dedicated for this purpose.

The DDL scripts in RDBMS can be seen as the Soft conceptual model, the database system as the Runtime conceptual model.

However, when applications are coded that access the database, the tables and names of tables are usually hard coded, and "expected" by the application, and therefore these applications do not conform to the principle of Conceptual independence.

6.5.3. A model for a complex IT system with Conceptual independence and Axiomatic design

If we combine Axiomatic design and Conceptual independence in an IT system we strive for replaceable components, for which each has the possibility to be adaptable to a specific Conceptual model. We outline this model in Figure 6.2. with Runtime components, each with its own functionality, such as: storage, exchange of data, access management, a rule engine. For specific functions, see Figure 6.3. These Runtime components all are based on the same meta model for conceptual models and can be configured with specific configuration languages.

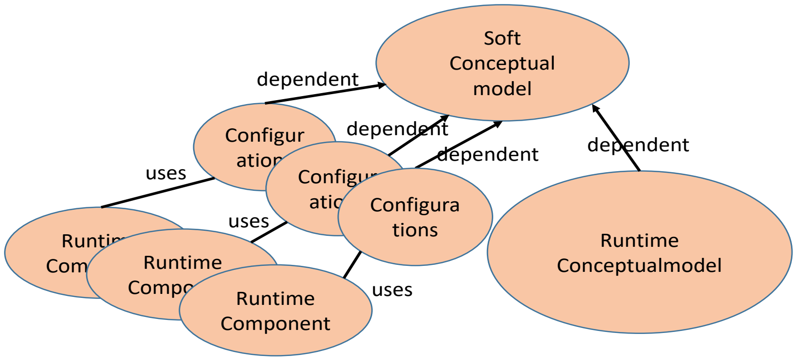


Fig. 6.2. Adaptable information system.

The Runtime components need a runtime Conceptual model such as a database system. A database system can be generated with a parser and generator system (not in diagram) without violating the Conceptual independence, see Fig. 6.3.

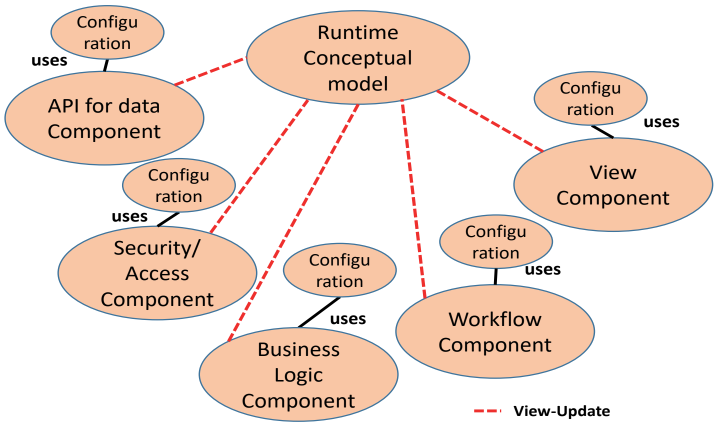


Fig. 6.3. Adaptable information system with system and business functions.

When the meta model, Soft conceptual model, the Runtime conceptual model and two Runtime components from Fig 6.3. are translated to a position in a Design matrix, we can map each independent Functional requirement to "one configuration" that adapts behaviour of the Runtime component. The configuration is then seen as a Design parameter. We will explain this in section 6.6.

6.6. Adding Conceptual independence to the Design matrix of Axiomatic design

6.6.1. Soft Conceptual model and Runtime Conceptual model added

We will describe an example of one general system function: “Show user interface for a specific Conceptual model” and one business function “Check consistency of data of one Conceptual model for an organization”.

In Table 6.2. the Design matrix is represented. For each general functional requirement the Independence axiom is applied, which is dependent on the metamodel. The Runtime is “generated code” that means it will give access to information in the Soft conceptual model. The parser and generator system has been left out of the Design matrix for simplicity.

Table 6.2. Design matrix, example.

	Functional requirement	**Design parameter**										
		Meta-model	DP1	DP2	DP3	DP4	DP5	DP6	DP7	DP8	DP9	DP10
	Base meta model	x										
FR1	Store conceptual model	x	x									
FR2	Give runtime access to conceptual model	x	x	x								
FR3	Run rules	x			x							
FR4	Check Business Rule 1	x	x		x	x						
FR5	Check Business Rule 2	x	x		x		x					
FR6	Check Business Rule 3	x	x		x			x				
FR7	Present user interface at Runtime	x							x			
FR8	Present GUI specification 1	x	x						x	x		
FR9	Present GUI specification 2	x	x						x		x	
FR10	Present GUI specification 3	x	x						x			x

We could add a third Runtime component in the Design matrix for other services for the business, that can be decoupled from the system, but not from the Conceptual model. This component can be easily added in the same manner as the other Runtime components. One can check that Runtimes are reusable in other systems based on the same metamodel.

Table 6.3. Descriptions of DPs.

DP	Description
DP1	Conceptual Model in text
DP2	Runtime Conceptual model
DP3	Rule engine
DP4	Configuration 1 of Rule engine
DP5	Configuration 2 of Rule engine
DP6	Configuration 3 of Rule engine
DP7	user interface Runtime component
DP8	Configuration of user interface1
DP9	Configuration of user interface1
DP10	Configuration of user interface1

Descriptions of the DPs are represented in Table 6.3. Here every DP is described and the mapping of FRs to DPs can be traced forward and backwards.

Adding Conceptual independence to system design with Axiomatic design solves the interdependence of the technical application on business information and hard coded names and terms.

6.6.2. Design parameters and Process variables

In Axiomatic design another Design matrix can be constructed in which dependencies of Design parameters and Process variables are represented. Process variables refer to the way DPs are generated and therefore concern choices of resources. In IT we thus observe the choices of programming languages, human resources such as IT developers and specific tooling needed to deploy the system in a technical way (Suh, 1998). In our example we have used a Metamodel for decoupling the Business terms from the technical implementation. It is disputable if this metamodel belongs to Design parameter or the Process variables. But, since it is essential for decoupling FRs we decide to explicitly include it in the first Design matrix in Table 6.2.

Table 6.4. Design matrix2, example.

DP	Description	Process Variable									
		PV1	PV2	PV3	PV4	PV5	PV6	PV7	PV8	PV9	PV10
DP1	Model in text	x									
DP2	Runtime Conceptual model		x								
DP3	Rule engine			x							
DP4	Configuration 1 of Rule engine				x	x					
DP5	Configuration 2 of Rule engine				x		x				
DP6	Configuration 3 of Rule engine				x			x			
DP7	User interface Runtime component								x		
DP8	Configuration of User interface1									x	x

Table 6.5. Design matrix2, example

PV	Description
PV1	Storage medium for Conceptual model
PV2	Database or Object base or other
PV3	Compiler 1
PV4	Syntax definition Rules
PV5	Rule1 in syntax
PV6	Rule2 in syntax
PV7	Rule3 in syntax
PV8	Compiler 2
PV9	Syntax definition UI
PV10	UI1 in syntax
PV11	UI2 in syntax
PV12	UI3 in syntax

However more can be said about the relation of DPs to PVs. In our example, we would need a medium for storage of the Conceptual model, compilers to generate Runtime components, a Runtime component that would provide access to the Conceptual model at runtime. For running the Rule engine and for presentation of the user interface, we need configurations in a syntax that is machine readable. Because we work with rules and screens that are independent of each other, independency of the configurations can be achieved. In Table 6.4. we show the PVs for the DPs in our example. The list of PVs is added in Table 6.5. For readability, we have left out DP9 and DP 10 in Table 6.4.

In Table 6.5 we find different systems for storage, that are independent of each other, different syntaxes for the Runtime components and configurations in specific syntax for rules or UI screens.

6.7. Conclusion

Researchers have found in case studies that interdependence of Business terms and technical application logic is an important factor that made IT systems hard to change. We argue that FRs are independent, but that dependencies are implemented in systems in DPs by coupling Business terms and application behavior. Separating the business terms from the system components by means of a Metamodel, a Business data model and using configurations, gives designers the ability to define independent DPs. The PVs that have been described in this paper, demonstrate that the IT developers already have access to the tooling and resources to construct a system based on Axiomatic design.

6.8. Discussion

The ideas stated in this paper are difficult to accept by the IT architecture research community because in technical papers in software engineering the problem does not exist. We have demonstrated that the problem does exist in the case studies, where business functionality is primary.

In IT architecture research a separation of the Conceptual model from application logic is often seen as a theoretical construct that is not applicable in practice. In practice however, it can be observed that commercial companies are developing systems that generate conceptual independent applications. Three commercial businesses in the Netherlands present themselves as software developers for adaptable systems (CGI Groep, 2016; Mendix, 2016; Thinkwise, 2016). A first analysis of these platforms demonstrates that Conceptual independence has been applied, without the explicit reference to this principle. We are currently studying the extent to which they have also implemented the Independence axiom, and the Information axiom. Research with regard to these aspects is in progress. Another case study is in preparation in which a real-life system in health care will be developed based on these principles.

7 Unintended effects of dependencies in source code on the flexibility of IT in organizations

This study links business requirements and adaptability of existing software systems. Organizations expect flexibility of IT with regard to business requirements. We hypothesize that the flexibility of business requirements is difficult in IT systems, because of software dependencies in the way domain knowledge is implemented. In this paper, we, therefore, explore how Business requirements have been implemented in the source code of three open source healthcare systems. Outcomes suggest that a tight interdependency of business terminology and functionality in source code hides business requirements from view and thereby hinders IT flexibility on higher levels.

7.1. Introduction

Scholars investigate strategic alignment of business and information technology (IT) for more than three decades within the information systems (IS) community. Recently, the importance of the role of flexible IT infrastructures for strategic alignment has been demonstrated anew for deployment, innovation, and evolution of IT systems in firms that operate in turbulent industries, including healthcare (Henderson & Venkatraman, 1993; Rogier Van de Wetering, Mikalef, & Helms, 2017; R Van de Wetering, Mikalef, & Pateli, 2017; Rogier Van de Wetering, Versendaal, & Walraven, 2018). In the software architecture domain, software adaptability is seen as a quality attribute of software in general. Thereby meaning, for instance, the applicability of technological innovations or new technical features. Software adaptability is not explicitly aimed at changes in the business domain (Bachmann et al., 2011; Bass, Clements, & Kazman, 2012; Tyree & Akerman, 2005). Expectations of the business do value general adapt-

This work was originally published as:

Tarenskeen, D., Van de Wetering, R., & Bakker, R. (2018). Unintended effects of dependencies in source code on the flexibility of IT in organizations. In *FedCSIS (Communication Papers)*, 87-94.

ability of systems, but also assume adaptability regarding business requirements. This current study focuses on changes in the business domain and business requirements and its consequences for software.

We examine adaptability of IT systems regarding business terminology and business requirements.

7.2. IT flexibility in enterprise architecture

Although Enterprise architecture methods such as TOGAF focus on high-level business requirements, in the Business architecture, they are meant to show relations between high-level requirements and supporting architectures. We use definitions of TOGAF, because TOGAF aims at describing all levels of the IT infrastructure. TOGAF describes the supporting IT as data architecture, application architecture, and technology architecture. The definition of architecture in this paper follows TOGAF's (TheOpenGroup, 2011): "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." Based on ISO/IEC 42010 according to TOGAF.

7.3. Related research on software evolution

Adaptability of software in empirical research can be positioned in the domain of research of Evolution of software. Within this domain, we notice that the evolution of business requirements is only marginally addressed.

Lehman strongly influences the research field of software evolution. The Laws of software evolution have been stated and evaluated during more than a decade of research (Herraiz, Rodriguez, Robles, & Gonzalez-Barahona, 2013; Lehman, 1996). Research in this field has made no explicit distinction between the evolution of systems based on requirements in general, and evolution of systems based on new business requirements.

Numerous studies have emphasized the complexity of source code changes after the initial system has been realized, for example, see (Ajienka, Capiluppi, & Counsell, 2017; Kagdi, Collard, & Maletic, 2007; Kagdi & Poshyvanyk, 2009). Studies that examine the relation of source code to IS architectures have a different focus than this research. They, e.g., aim at developing frameworks for software architecture evolution knowledge (Ahmad, Jamshidi, & Pahl, 2013), or frameworks for classifying architecture-centric software evolution research (Jamshidi, Ghafari, Ahmad, & Pahl, 2013), or on automatically updating architecture documents based on software changes (Haitzer, Navarro, & Zdun, 2017).

7.4. Axiomatic design and conceptual independence

To present our point of view, we start by explaining the theoretical basis for adaptable and flexible low-level software components in an IT architecture. The theoretical views focus specifically on the adaptability of business requirements instead of on adaptability of software in general. The theoretical principles of Conceptual independence (CI), and the independence of functional requirements such as described in Axiomatic design (AD) (McGinnes, 2011; McGinnes & Kapros, 2015; Suh, 2001) will be researched in this study in real-life software.

We report on a code mining study of open source code for Healthcare organizations for electronic health record systems (EHR), to examine the way business terminology is applied. Then we will argue that there is a direct link of adaptability on the source code level to IT flexibility as expected by the Business architects.

7.5. Research questions

We explore three selected open source software systems to find out if a separation of business terminology and application code has been effectuated, to create flexibility in the source code (CI). Hence, our first question concerns description of the software systems by the developers:

RQ1: Are indications of CI found in the documentation?

Then, we question the interdependency of the data model and the application source code. We implicitly assume that the data in the database model will represent the data that will be persistently stored.

RQ2: Does code demonstrate interdependency of table names and source code?

Next, we want to examine the flexibility of the specific healthcare terminology in the software system, based on CI. Thus, we define:

RQ3: Is CI applied in the software application?

Next, for AD functional requirements are primary. So we define:

RQ4: Does the source code of the system show different components that are related to separate Functional requirements?

The remainder of this paper is structured as follows. First, we highlight the theoretical aspects relevant to this study. Next, we present our methods section which is followed by the results section. We then discuss our findings and end with concluding remarks and some suggestions for future research.

7.6. Theoretical background

7.6.1. Axiomatic design

The principles of AD are explained by Suh (Suh, 2001). He explains how a design method should account for the independence of functional requirements and a low information density in different design parameters of the system. He calls these characteristics the Independence axiom and the Information axiom. The systems he describes are industrial systems, but he emphasizes that these principles can be applied to IT (Suh, 2012). We interpret design parameters as design components of an IT system. The objective of AD is to realize systems that are flexible and understandable. With AD, the designs are iteratively developed and have the domain of customer needs, demands and requirements as a point of departure. From customer needs, functional requirements are inferred. In the design, the functional requirements are formulated independently from each other and can be changed in the design without affecting the rest of the design. The principle of Independence of functional requirements is at the fundament of AD and can be compared to patterns in software engineering (Larman, 2005), such as separation of concerns. However, realizing independent functional requirements is not a priority in software engineering (Buschmann, Henney, & Schmidt, 2007a).

7.6.2. Conceptual independence

We advocate CI, the decoupling of healthcare terminology and domain models from software code to be able to alter healthcare terminology or domain models flexibly. We ground this choice on previous case studies that argue that AD principles are hard to implement in software systems, because of the interdependence of data models and the behavior of the system (Tarenskeen & Bakker, 2017). The interdependency of data models and application code has been extensively studied in IS research (Aryani, Perin, Lungu, Mahmood, & Nierstrasz, 2011; Cleve, Gobert, Meurice, Maes, & Weber, 2015; Mens et al., 2017; Qiu, Li, & Su, 2013).

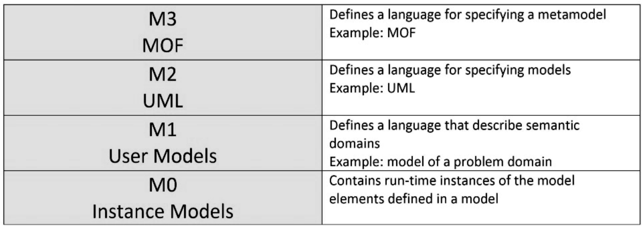
McGinnes points to the interdependence of data models and software application code as Conceptual dependence. He advises the decoupling of the data model and application functionality, meaning the behavior of the application (McGinnes & Kapros, 2015). McGinnes defines the Conceptual model as the structure of the information that is used in the business. Comparing the Conceptual model to the Domain model in UML, we find that in UML often behavior is added to the classes in the Domain, this is not the case in McGinnes' Conceptual model.

McGinnes adds behavior to Concepts by ordering Concepts in Archetypical categories, that applications can access. The applications have a responsibility to interpret the Archetypical categories. The applications add the behavior based on the specific Archetypical category. For instance, for "Location" the application knows that the instances of this category can be presented on a map.

7.6.3. Relation of Conceptual model to model-driven development

McGinnes describes the conceptual model as a (business) data structure that is used by the application. The metamodel, of the conceptual model, is fixed, the content is variable. These structures are comparable to MDD described by the OMG, Object management Group (Object Management Group, 2016). There are four levels of models, each function as a meta-level of the lower level. These levels are M0 to M3. McGinnes positions the conceptual model itself on level M1 as data.

We will address the meaning of these levels briefly.



M3 MOF	Defines a language for specifying a metamodel Example: MOF
M2 UML	Defines a language for specifying models Example: UML
M1 User Models	Defines a language that describe semantic domains Example: model of a problem domain
M0 Instance Models	Contains run-time instances of the model elements defined in a model

Fig. 7.1 Diagram of Modeling levels of OMG

The M1 level is the most important and most discussed modeling level in practice of software engineering. It shows categories or classes and the associations between them. The content consists of terms, for example, Person, Product, Order. The level is comparable with table names in RDBMSs. The model M0, on the lowest level, contains the instances of categories M1 stored. It is comparable to records in RDBMSs or instances of objects in programming languages. See Figure 7.1., published in a whitepaper explaining the different levels of Modeling of OMG (Object Management Group, 2017).

The M2 level contains the description of model elements in a modeling approach; this is the meta-level of the description of, e.g., UML-models. The highest level (M3) contains a description of all (possible) modeling approaches. It is intended for comparing different modeling approaches (Object Management Group, 2017).

The description of McGinnes of the model is at the M2 level, the model that concerns business concept types is an M1 level-model (Bézivin & Gerbé, 2001). In this paper, we will not explore the similarities of OMG and McGinnes further. We think the challenge in system development lies in separating Conceptual models from behavior.

We argue based on ideas behind Conceptual dependence that CI is a prerequisite to being able to separate the different functional requirements from each other in the behavior part of the application (Tarenskeen & Bakker, 2017).

7.6.4. Ampersand as illustration

We first, will describe a prototype system called Ampersand[2], based on a requirements specification language with relational semantics to illustrate the feasibility of implementing principles in source code (G. Michels, S. Joosten, J. Woude, & S. Joosten, 2011a). Ampersand relies on model-driven development (MDD) to generate systems entirely defined by its domain model and business rules.

7.6.5. Ampersand applies CI and relation algebra for AD

We will explain the workings of Ampersand to demonstrate that flexibility of business functional requirements is feasible on source code level. This example is added for technical readers to explain the low-level code involved in separating business terms from application code. It also demonstrates with low-level code that a possibility to separate the business requirements from each other can be accomplished. The system Ampersand has separated the conceptual model from behavior. It, therefore, conforms to CI. It is based on relation algebra and has a mathematical structure (Michels, et al., 2011a; G. Michels, S. Joosten, J. v. d. Woude, & S. Joosten, 2011b). The conceptual model in Ampersand consists of concepts and relations between concepts. All information about concepts and relations is described in an Ampersand script (typically a .txt file). There is no extra information of the business hidden in the software system. Behavior is described and defined in invariant (or declarative) business rules. The behavior is only applicable to the concepts and relations in the script. A script contains one Context that is entirely separate from other Contexts that can be defined in Ampersand. Ampersand applies rules as a way to connect the conceptual model structure to behavior. Rules can also be defined to check the consistency of data. Ampersand applies Rule checking behavior to define the software behavior. Rule checking is applied to Concepts and Relations in the Ampersand model. Examples from rules in healthcare can be: Diagnoses must have a relation to a Medical doctor, Diagnoses must have a date, a Patient cannot receive medication without the consent of the MD.

Each rule must be independent of the other rules in Ampersand, and therefore, behavior can be defined according to independent business functions.

7.6.6. Ampersand Runtime

The Ampersand Runtime can read, parse the script and import the Conceptual model, data, and rules. The script contains models on level M1 and M0. After reading this, a Rule engine checks business rules and signals violations. It operates on any script that conforms to the syntax and constraints of the Ampersand approach (On level M2).

7.6.7. Example Ampersand script

The following description of the Ampersand script is the model in natural language on level M2 of the OMG. Here we describe constraints and model elements (categories)

2 Named after the ampersand symbol (&). According to Michels et al. the name refers to getting the best from both business and IT, i.e., achieving results from theory and practice alike, and realizing the desired results effectively and more efficiently than ever before.

that can be present in the script.

The first term in the Ampersand script is the word: CONTEXT. It signals the beginning of the script. ENDCONTEXT signals the ending. Then a PATTERN is presented consisting of CONCEPTS and RELATIONS.

After the pattern, the word ENDPATTERN closes this part, and in the script, PROCESSs can be defined regarding Ampersand RULES.

Summarizing, we can state that Ampersand follows the principles of CI by providing flexibility for the structure and naming of the data model. There are two different methods for keeping the conceptual model separate from the application code in Ampersand. First, the Ampersand Runtime works directly with the script and does not know about the domain in the script. Second, the script can be used for MDD. The Ampersand system conforms to the Independence axiom of AD, at least as far as functional requirements are concerned that can be defined in rules.

We have explained the workings of Ampersand in detail to demonstrate that flexibility of business functional requirements is feasible on source code level.

7.7. Research method

7.7.1. Data collection procedure

We report the outcomes of code analysis of three systems. Two of these systems are frequently used in international health practice. The third system implements the standard of openEHR; a development we see more often these days. The latter claims to support different kinds of models for medical data. The research data are downloaded systems from GitHub. These were run locally to assess runtime dependencies and check if the source code is complete. Then, we have analyzed the documentation and the source code. We classified source code in types, the source code for libraries, the source code for initializing the database, source code for user interface frameworks and source code for business and other functionality in the software system. Only the last type of files have been examined in RQ2.

Since the idea of the paper is to evaluate open source systems in healthcare for application of CI and AD, we have searched for open source systems with an active community. The systems have been included in the references (websites and date) (openEMR, 2017; openMRS, 2017a, 2017b; Pazos, 2017). All of these are web applications that were run by us with an apache or tomcat server. Cabolabs is written in grails, openEMR in PHP and openMRS in java. All could operate with a MySQL database. The cabolabs openEHR download consists of 1351 files with 48 different extensions. The openEMR consisted of 12118 files with 130 extensions. The openMRS software has two downloads, the standalone consists of 723 files with 56 extensions. Because we also wanted to analyze the java code, we have also downloaded the core of openMRS with 1623 files with 40 different extensions.

7.7.2. A multistep approach

For each system, we applied a multistep approach including the following action: (1) running the systems locally, (2) analyzing all relevant and available documentation, (3) analyzing the directory structure, (4) extracting the data model from the MySQL database, (5) analyzing specific healthcare terms, (6) analyzing if database tables are hardcoded or generated for the particular system and (7) selecting of source code with (business) functionality (manually).

We incorporate specific methodological considerations and action (per research question) into the results sections.

7.8. Separate analysis of healthcare terms

We wanted to assess if application code depended upon hardcoded names in software code derived from healthcare. In the source code distinguishing between names that refer to healthcare terms and names that are necessary to follow the technical program flow, is difficult. Since we do not have expertise on healthcare terminology, but we wanted to signal the terms that were healthcare terms, we have asked independent reviewers and one healthcare professional (psychiatrist) to review the names of database tables and list the names derived from or partly related to healthcare.

We have asked four reviewers that are not researchers or in any way related to the case studies. We have asked them to evaluate every table name in the three databases.

Table 7.1 Evaluation of healthcare related terms in table

		openEHR	**openEMR**	**openMRS**
Number of tables		59	212	148
Number of tables with Health care related names	According to 3 of 4 reviewers	1	30	16
	According to Healthcare professional	2	62	50
Proportion of tables with Healthcare related names	According to 3 of 4 reviewers	0,02	0,14	0,11
	According to Healthcare professional	0,03	0,29	0,34

The scores of the healthcare professional have been registered separately also. For every open source program, there were two scores: the percentage of table names that three of the four reviewers labeled as healthcare related. Moreover, the separate score of the healthcare professional.

In Table 7.1. we find the number and percentage of table names with recognizable healthcare terminology parts. The healthcare professional classified more names as healthcare related than the other persons, but all the table names that the 3 out of 4 persons listed were a subset of the names that the healthcare professional listed. We are now able to assess interconnectedness of application code to hardcoded healthcare terms.

7.9. Results conceptual independence and axiomatic design in source code

7.9.1. Conceptual independence in the documentation

This section addresses RQ1. We have extensively read the associated documentation and searched for indications that the system is adaptable based on healthcare terminology. Through our analyses, and also based on our review of the Information-model of openEHR on the website of openEHR, we conclude that openEHR indicates CI. A quote from documentation of openEHR confirms this view (openEHR Foundation, 2020): "*Your EHR system does not need to know a priori about any of the clinical data it will process, such as vital signs, diagnoses or orders. Models for those things are developed separately. Models for data sets and forms are also developed separately, and UI form components are generated from these definitions.*"

The data structure is said to be very flexible and can support transformations to other healthcare terminology standards.

In openEMR, there exists no reference to a model, but we find a description of the Database structure (openEMR, 2014). There is some variability for the conceptual model, by which we mean, the user can define categories in one table. Thus, openEMR is partly flexible concerning terminology, but not concerning models of healthcare data. Finally, openMRS shows signs of CI and AD. To highlight this particular view, we quote from the wiki documentation of openMRS (Mamlin & Jindal, 2017): *"At the heart of OpenMRS is a concept dictionary. This dictionary, much like a typical dictionary, defines all of the unique concepts (both questions and answers) used throughout the system.*"

The software itself, openMRS, in essence, is constructed to support 'modules.' Implementations can modify the behavior of the system to meet local requirements using these modules. Because changes can be added to the Conceptual model, it is not necessary for everyone having to agree on a single approach.

7.9.2. Interdependency of table names and source code

We now report the results of for RQ2. These results consist of totals of code mining results.

We have defined two indications of the interdependency of code and data, i.e., I) hardcoded use of table names in the source code of more than 80 percent of the tables and II) hardcoded use of table names in the source code of more than 80 percent of the source code.

We did not find the expected first indication in every system. We would have expected the use of all the tables in the source code. If table names are missing, they are not used. Since the named applications are the only applications that use the database tables, this needs further investigation.

The second indication has been signaled in the source code files. If names of database tables in source files are hardcoded, then any particular change in table names implies changes in the source code. With table names spread over different files, then changes in table names lead to changesin multiple maybe interdependent files, leading tounpredictable behavior of the software application.

Why would a table name change? If ideas about the Conceptual model change or if other functionality needs other data concepts or maybe extra attributes (columns in tables) then the table names and columns will change. Adaptations of concepts and table names or columns are frequent in the evolution of code (Qiu, et al., 2013). In this empirical study of Qiu, it was found that adding tables, columns and changing names of tables and columns frequently appear. We focus here, specifically, on the names that are healthcare related, because we signal a relation between business terms and source code files.

Our method can be described in the following way: we have extracted all table names from the applied database management system and have counted the number of times these names are hardcoded in software source code. We have calculated the percentage of database table names that were found in the source code files. We have counted the number of source code files that access database table names directly or use Class names that are derived from table names, for instance by removing the dash. In Fig.7.2. the relations of the source code files to table names are visually presented. We also have calculated the percentage of source code files, that access table names directly.

Counting will demonstrate the relation.

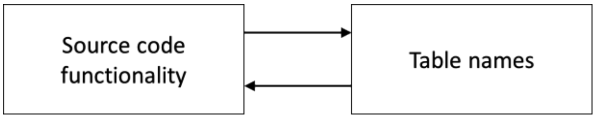


Fig. 7.2. The existence of relations between source code and table names

Concerning openEHR, 46% of table names have been found in the programming code, but only two of those are marked as Healthcare related. The names are doctor-proxy and patient-proxy, but no table names are related to medical knowledge. The groovy files with table names accounted for 99% of 176 files, but these were not marked as Healthcare related, exception above. In groovy files, 69% class names have been found, that are derived from table names. Groovy files with these class names accounted for 69% of the groovy files.

In the openEMR download, 82% of table names are found hardcoded in PHP-code. Including 29 of 30 with Healthcare related table names. In the PHP-files 94% of 5401 files have access to hard-coded table names.

In the openMRS-core download, we have found 57% of table names are hardcoded in java-source code. Including almost all table names (14 of the 16), that have been marked Healthcare related. In the Java-files in the openMRS-core, 100% of 1019 files access hardcoded table names.

Based on the second indication, we find an extensive interdependency between source code and database table names in all three systems.

7.10. Is conceptual independence applied in the software application

This paragraph reports results for RQ3. The indications below are derived from characteristics of CI:

- *Indication: No hardcoded use of healthcare-related table names in the source code.*
- *Indication: A presence of a separate structured model for healthcare terms, in the source code for generating database tables.*
- *Indication: A presence of a separate structured model for healthcare terms, in a separate file.*

We expect that when the healthcare-related table names are found in source code files, then changes in healthcare terminology directly affect the source code.

So for the first indication of RQ3, we have to mark database table names that have a direct reference to the medical terminology in the system. For this indication, we had first to classify all table names in “Unknown name” and “Healthcare terminology name.” See section 7.8. Then we searched for occurrences of healthcare-related table names in source code files. Two systems: openEMR and openMRS, applied hardcoded healthcare-related table names in the source code.

The other two indications, above, are meant to demonstrate a separation of the conceptual model and the application code, as is a characteristic of CI. In detail, we have found that openEMR and openMRS have applied frameworks for separating business domain terms (the Conceptual model) and business logic from the rest of the application code. The frameworks used are Zend for openEMR and Hibernate for java in openMRS. These frameworks and the related source code of the systems have been analyzed. The frameworks use script code for defining the (Business) Conceptual model. They do not separate the Conceptual model from the behavior of the software. Therefore these do not comply with CI. The frameworks aid the developers with building and partly generating source code. Hibernate helps developers in separating database management systems from source code but does not aid in decoupling business terminology from source code. The framework script code then becomes part of the source code. We cannot directly extract the applied Conceptual models.

The software of openEHR contains separable Conceptual models apart from application logic. We confirm the existence of separate Conceptual models because we also find “parsers” and “indexers” in the source code.

For the last indication, we have counted the number of times table names can be found in one file, to search for indications of a definition file for the Conceptual model. In openEHR, we found the “opt-file” and “adl-file,”, in which the M1 model is included as data. They comply with the M2 model of openEHR. Therefore it can be used for separating the Conceptual model from the behavior of the software.

In the openMRS source, the liquibase tool is applied for updating tables based on changes in the database for new modules. With the liquibase functionality updates on database structure and data can be automated with liquibase.xml-files. The M1 model is input as xml-data, but no M2 model can be found.

In openEMR, only an .sql file was found that contained 188 of the 212 tables. The healthcare related terms are not included as data but are hardcoded in sql. It cannot be used as an M1 model, because changing it will break the source code and no M2 model can be found.

Concluding: In the source code of Cabolabs openEHR server system all three indications have been found. Several files with the Conceptual model and its instances (M1 and M0) have been signaled. These files with extensions .adl and .opt can be reused by other openEHR standard based systems. Cabolabs openEHR-server applies CI.

In the other systems frameworks such as Hibernate and Zend have been used, for partly separating the model from the application code. Further, the frameworks do not distinguish business terms from application code classes. The consequence is that the current application of frameworks involves code programmers for adaptation of business logic and business terminology.

7.11. Axiomatic design applied in source code

In this paragraph, a report of RQ4 is given. For AD, functional requirements are primary. In AD it is required that the software system can be divided into components that are related to functional requirements. Systems based on AD will be adaptable based on changing functional requirements because business IT architects can pinpoint specific source files where changes are necessary.

RQ4 will lead to demarcation lines in the Runtime components or demarcation lines in the source code, which has different independent functional requirements.

- *Indication: Existence of directory structures in the source code that show Functional requirements*
- *Indication: Existence of runtime modules that can be added and deleted for Functional requirements behavior that is executed*

The indications for Axiomatic design will be studied in detail in future research. In this overall check of the source code, it is found that openMRS contains a directory structure for separate modules. We find complementary functionality in the openMRS runtime application because modules with Business functionality can be turned off and on. With the openEHR server software, tooling is under construction that can generate User interfaces based on the opt-files. Moreover, thus separation of high-level functional requirements can be realized.

7.12. Conclusion and discussions

In this study, we have explored the adaptability of source code concerning the business requirements and changes in the business domain terminology. Interdependency of the data model and the application code can make systems hard to change, this is seen in the literature and in our investigation of open source healthcare systems.

However, the dependency of the application source code on healthcare terms can be avoided by separating these terms in a separate model as input for the application. We demonstrate this with the Ampersand prototype, where indications for CI and AD can be located in the source code.

We have explored how business terms and business functionality appear in application source code in three open source systems actively used in healthcare. We have shown that CI, separating the business terms from the application software, can be applied and is applied in openEHR and partly in openMRS. AD has not been studied extensively in this case, but indications for AD are found in openMRS.

We conclude that because of the extensive interdependence of the data model and application source code in openEMR and openMRS, business terminology becomes part of the source code and cannot be adapted without radically changing the source code. So we conclude that in these systems the flexibility of business terminology is obstructed if the business terminology is not explicitly separated from the application source code.

Despite this studies contributions, there are several limitations that future research should address. The researchers remark that an alternative to separating the conceptual model from application source code would be to use tooling for source code editing based on business requirements. Moreover, currently, some indications for this kind of tools were present in the source code that is examined. Frameworks help to separate the Conceptual model from the application code, but in the end, Conceptual models become an integrated part of the source code. When the Conceptual model is included in the source code, then it will depend on professional skills or discipline of the programmer(s), to check that the Conceptual model will stay separated from application code. Since frameworks do not distinguish health care terms from software application classes, medical expertise is necessary to locate these.

In this paper, we have only studied software architecture for a limited number of applications and components. Therefore it can be questioned if a full-scale application of this principle can be implemented in enterprise architectures. We are currently researching the design and implementation of a separate (conceptual model-layer) data layer in a large-scale healthcare IT architecture.

8 The Contribution of Conceptual Independence to IT Infrastructure Flexibility: The Case of openEHR

In the Netherlands demands on IT support in healthcare organizations are increasing. New visions on healthcare focus on patient-centered healthcare, where mutual consultation among healthcare professionals in the network becomes a standard process. Recent governmental regulations prescribe that patients must be able to access personal health records. IT flexibility is needed to allow organizations to meet new demands. In this study we focus on Conceptual Independence (CI) because CI, as a design principle, can improve the adaptability of Information Systems (IS). Software with CI operates on flexible data models that are independent of the CI based application. Therefore, it is claimed that a standalone IS becomes more flexible with CI. We extend the claim by demonstrating that CI affects the flexibility of the entire IT infrastructure. We investigate which dimensions of IT flexibility are responsible for the improvement. Multi-case study research has been performed following a mixed-methods approach in 10 mental healthcare organizations. Five have implemented openEHR, a proxy for CI, and five have not. Data has been collected with a questionnaire of IT infrastructure flexibility and semi-structured interviews. The data synthesis shows a positive effect of CI on IT flexibility, as CI increases the adaptability of IS, transparency and standardization of the IT infrastructure.

8.1. Introduction

In the Netherlands, as in other European countries, due to demographic changes and advances in technology, new strategies and visions on healthcare emerge. Healthcare is making a shift to ward patient-centered care and this has a substantial impact on how healthcare organizations work and engage with their patients and clients. As the

This work was originally published as:
Tarenskeen, D., van de Wetering, R., Bakker, R., & Brinkkemper, S. (2020). The Contribution of Conceptual Independence to IT Infrastructure Flexibility: The Case of openEHR. *Health Policy and Technology, 9*(2), 233-244. doi: https://doi.org/10.1016/j.hlpt.2020.04.001

digitalization invades all aspects of healthcare, providers need to leverage their current and future IT investments genuinely and design applications that are future-proof and adaptable to continuously changing requirements to match clinical and administrative processes (Rogier Van de Wetering, Versendaal, & Walraven, 2018). The extant literature tends to study IT applications in infrastructure as a 'black box', meaning that no particular reference is made to software or software structures or other components of the IT infrastructure. We describe IT applications as software for end users and Information Systems (IS) as a special type of IT applications, namely data-intensive IT applications. IT infrastructure is the whole of configurations of interlinked systems and IT applications in the organization. The contribution of this research is to examine the relation of software characteristics in the black box of IS to IT flexibility.

Scholars generally refer to modularity as a characteristic of flexibility to define flexibility in IT infrastructures (Baldwin & Clark, 2006; De Reuver, Sørensen, & Basole, 2018; Schilling, 2000). However, in practice, inflexibility still exists in critical applications that rely on extensive data, such as in Electronic Health Record (EHR) systems, especially when data has to be exchanged with other IS. Synthesizing from the literature it appears that combining the functionality of IT applications has not been self-evident. According to Bygstad and others (Bygstad, Hanseth, & Truong Le, 2015) silos in IT infrastructure obstruct its flexibility. These silos came into existence as past adaptations of IT applications to a bureaucratic way of working. Restructuring silos in current systems is difficult.

EHR systems store patient health information, such as laboratory results, medication lists and allergies. EHR systems allow doctors to work more efficiently and drive standardized work practices across the care continuum. However, we find that maintainability is problematic (Atalag, Yang, & Warren, 2014). We presume that conceptual models play a crucial role in maintainability. Working with conceptual models is problematic because of the complexity of medical terminology (Rector, 1999). Ignoring the conceptual models in the design of IS leads to heterogeneous, incompatible and inflexible conceptual structures (McGinnes, 2011). McGinnes and Kapros describe the principle of Conceptual Independence (CI) (McGinnes & Kapros, 2015). The principle is defined as an alternative approach for design of IS with the objective to create flexibility in IS. They claim that CI, as a decoupling of medical conceptual models and application code, will lead to flexible IS. We enlarge the scope of CI and state that CI will increase IT infrastructure flexibility as a whole.

This study addresses the following research question: *"How do implementations of CI in IS lead to an increase in flexibility in organization-wide IT infrastructure?"*

There have not been large implementations of CI in real-life systems since the publication in 2015 (McGinnes & Kapros, 2015). Therefore, research about CI implementations affecting IT infrastructures has not been executed before.

Looking for an alternative we have decided to study a proxy, an openEHR implementation, based on similar characteristics. The openEHR is an open, flexible standard applied for modeling medical knowledge and information about patients (openEHR Foundation, 2020). Hence the underlying openEHR software is based on CI, as we will argue in par. 2.5. There has been one large study of openEHR and infrastructure in Norway by Ulriksen et al. (Ulriksen, Pedersen, & Ellingsen, 2017) which concludes that information structure and installed base have to evolve together. However, these scholars do not explain which characteristics of the installed base could facilitate or obstruct IT flexibility. We fill this gap in the literature by explaining the role of the underlying software design.

We need real-life data and therefore apply a multiple case study approach with mixed-methods research.

8.2. Theoretical background

8.2.1. IT Infrastructure Flexibility

The concept of IT flexibility, in essence, refers to a shareable and reusable architecture that enables IT resources to develop and adapt system components quickly. It allows organizations to anticipate new market opportunities as business conditions change (Kim, Shin, Kim, & Lee, 2011; Tallon & Pinsonneault, 2011). IT flexibility emerges as a crucial ingredient of the deployment of digital business strategies (Overby, Bharadwaj, & Sambamurthy, 2006; Sambamurthy, Bharadwaj, & Grover, 2003). IT infrastructure flexibility has been confirmed as an influence on IT flexibility (Kemena, Wetering, & Kusters, 2019). Van de Wetering and others describe the importance of IT flexibility in its relation to dynamic capabilities (Rogier Van de Wetering, Mikalef, & Pateli, 2018; Rogier Van de Wetering, Versendaal, et al., 2018).

Van de Wetering, Mikalef and Pateli (R Van de Wetering, Mikalef, & Pateli, 2017) define IT flexibility as "the degree of decomposition of an organization's IT portfolio into loosely coupled subsystems that communicate through standardized interfaces". We apply this definition. The definition conforms to previous studies by Byrd and Turner and Duncan(Byrd & Turner, 2000; Duncan, 1995). Previous scholarship unfolded some of the key qualities that comprise IT infrastructure flexibility. These qualities are modularity, standardization, transparency and scalability (Byrd & Turner, 2000; Duncan, 1995; Tafti, Mithas, & Krishnan, 2013). The complementary effect of these synthesized qualities enables organizations to adapt and co-evolve with the changing conditions (Mikalef, Pateli, & Van de Wetering, 2016). These dimensions can be defined in the following way, modularity, "Loose coupling" is the main idea behind modularity. With the isolation of independent components, it will be easier to replace and adapt single parts in the IT infrastructure. With transparency IT systems will behave as one integrated system with seamless accessibility to functions and data. Flexibility in this sense gives end users access to different elements in the infrastructure.

For standardization, we observe that organizations apply comprehensive standards for hardware and software. The last dimension scalability measures how well the IT infrastructure can be scaled up and upgraded when adaptation is necessary due to growing demand and an increasing number of users (Mikalef, et al., 2016).

8.2.2. Conceptual models are crucial in Information Systems

Conceptual modeling has been noted as a bottleneck, specifically in healthcare, where medical specialists prefer free texts for registration of medical data (Cios & Moore, 2002). In healthcare domain knowledge consists of medical knowledge. The representation of domain knowledge in conceptual models is necessary for developers of EHR systems. If, due to a growing number of independent conceptual models, the exchange of data becomes difficult, we see conceptual incompatibility (McGinnes, 2011). Rector (Rector, 1999) states that difficulties of terminology in patient systems in healthcare have been underestimated and that this problem leads to specific issues in software for patient systems. He mentions ten topics where misunderstandings can arise, such as interpretations of medical specialists and observed test results.

8.2.3. Design principle for Conceptual models in CI

The role of inflexibility of conceptual models is often mentioned in relation to reuse of functionality. Functionality can be reused without reprogramming the application in software. There has been a large number of studies showing the difficulties in reusing functionality, because of the low-level interdependence of data structures and application code (Garlan, Allen, & Ockerbloom, 1995, 2009; Lehman, 1996; Qiu, Li, & Su, 2013). Expectations of Service Oriented Architecture (SOA) to enable reuse were high. However, SOA did not solve the reuse issues, as Joachim demonstrates (Joachim, Beimborn, & Weitzel, 2013). McGinnes and Kapros (McGinnes & Kapros, 2015) tried to find a solution by separating conceptual models in the software application from the application logic. They argue that the separation of conceptual models leads to an Adaptive IS (AIS). The AIS is a system that can sup port any conceptual model. Therefore, CI can improve reuse and allow organizations to adapt conceptual models in IS without requiring re-programming. They describe six principles that are sufficient to achieve the separation in Table 8.1. (McGinnes & Kapros, 2015). When conceptual models have been decoupled from code, Principle 1, the software contains functionality that can be reused with all conceptual models based on a meta-standard. Conceptual models conform to a standard that is understandable and machine readable. Principle 2 states that archetypical categories can initiate appropriate semantic behavior based on categories of entities, such as functionality for showing the instances of the category "Location" on a map.

If the software can operate on multiple instances of conceptual models simultaneously (Principle 3) then, the software must be able to correctly identify data belonging to conceptual models (Principle 5 and 6). Entities must be uniquely identified independent of the conceptual models (Principle 5), checks and constraints are enforced at input (Principle 4) and all data is correctly labeled (Principle 6). Thus, the principles occur together in AIS.

8.2.4. Conceptual Independence is a modeling approach

The design principle of CI in modeling IS can be positioned in software engineering practice, similar to Model-driven design, as described by the Object Management Group (OMG)(Object Management Group, 2017). The OMG distinguishes different layers or levels. An overview of the levels is shown in Table 8.2. (Object Management Group, 2016). There are four levels of models (M0-M3), in different degrees of abstraction. The content of the models cannot be determined by the software systems but has to be input by human modelers. The meta-model of CI is positioned on M2. Analogous to Model-Driven Development (MDD), we conclude that there has to be a meta-model for conceptual models in the software.

CI, however, is not the same as MDD. MDD aims at modeling software systems, data models and behavior to generate the complete software code from its model. It does not per se separate conceptual models and application code in resulting code.

8.2.5. OpenEHR as an implementation of CI in the medical domain

We resort to openEHR implementations in this study, because we have no access to conceptual models in software in existing systems, except for open-source software (Tarenskeen, Van de Wetering, & Bakker, 2018). We argue that the openEHR implementations can be approached as a proxy for CI. In openEHR flexible, extendable conceptual models exist and are termed archetypes. The archetypes have been decoupled from application logic, as in the general characteristic of CI. In openEHR the decoupling is termed two-level-modeling, where medical knowledge is decoupled from run-time knowledge and patient information (Beale & Heard, 2007; openEHR Foundation, 2020). In theory according to Beale (Beale, 2002; Beale & Heard, 2007), archetypes are representations of medical concepts formulated in a meta-standard, Architecture Description Language (ADL). The same IS can be used with different archetypes (CI Principle 1). Archetypes can be categorized in such a way that specific user interface presentations can be initiated (CI Principle 2). Conceptual models, as archetypes in different versions, can co-exist in one application (CI Principle 3). Principles 4-6 are present in openEHR software, such as checks on the consistency of data with archetypes (CI Principle 4) and registration of data with the different archetypes (CI Principle 5 and 6). A fundamental difference between openEHR and CI is, that CI describes principles for software design and openEHR specifies a meta-standard for conceptual models in medical domains. Therefore, we characterize CI as the underlying software design for openEHR in medical domains.

8.3. Research framework and propositions

8.3.1. Influence of CI on IT infrastructure flexibility

In this section, we formulate propositions for possible effects of CI on IT infrastructure flexibility. Usually, the six principles of CI are not observable by functional management and users as the principles have been seamlessly integrated. For discussing CI in interviews we need observable characteristics.

The observable characteristics are derived from the text of the main paper of McGinnes and Kapros about CI (McGinnes & Kapros, 2015). CI will lead to an Adaptive

IS (AIS). An AIS is an IS in which the conceptual model can be adapted without reprogramming the IS. The AIS is an observable characteristic (I). Secondly, we have analyzed the 6 principles of CI in detail. Principles 1 and 2 from Table 8.1. emphasize the reusability of functionality in systems with CI. "Reusable functionality" is visible (II). Thirdly, in Principle 3, the multiple conceptual models in one IS can co-exist. According to the levels of the meta-model of OMG, in Table 8.2., there has to be an observable meta-model for conceptual models, that can be distinguished (III). Finally, Principles 4-6 are not directly observable. We have argued in par 2.3 that the principles must exist in a working system with CI. The research framework in Fig. 8.1. links the different observable characteristics of CI to the dimensions of IT infrastructure flexibility. Overall, we expect CI to increase IT infrastructure flexibility. We formulate one or two propositions for every dimension of IT infrastructure flexibility.

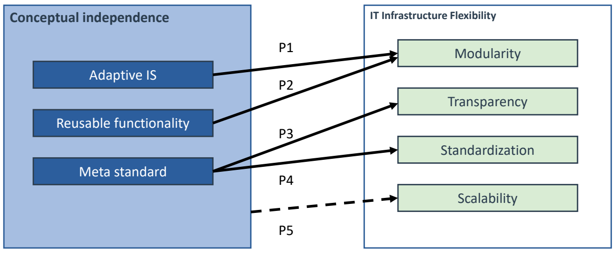


Fig. 8.1. Framework of the effects of CI on IT infrastructure

8.3.1.1. CI and modularity

McGinnes and Kapros (McGinnes & Kapros, 2015) state that, if CI is implemented in a single IS, it is called an adaptive IS (AIS). We expect that if this AIS is positioned centrally in the IT infrastructure, the flexibility of the whole IT infrastructure will be enhanced because it will break open the silos, as described by Bygstad et al (Bygstad, et al., 2015). Proprietary inflexible conceptual models in IT silos hinder IT infrastructure flexibility, because providers are dependent on the software vendor to make functional changes to the applications. And vendors are typically inclined to postpone changes, because they affect all users in other client organizations. CI can counteract the consequences of proprietary models. Hence, we define:

> **Proposition 1.** *Decoupling the conceptual models from application logic increases modularity in IS, therefore increases the modularity in organizational IT infrastructure.*

The concept of modularity is based on ideas of Schilling (Schilling, 2000), Simon (Simon, 1962), Baldwin and Clark (Baldwin & Clark, 2006) and Byrd and Turner (Byrd & Turner, 2000). Modularity, defined as a structure of loosely coupled independent sub-systems, will be a primary factor in the ease of changing software systems in general. We propose that reuse of functions enables isolating the functionality in software. Therefore, modularity will improve with CI and we define the following:

Proposition 2. *Because CI facilitates reuse of functionality based on the same meta-model, decoupling functionalities will be possible, not only in one IS, but in the whole IT infrastructure, thereby increasing the modularity of organizational IT infrastructures.*

Table 8.1. Principles of Conceptual Independence (McGinnes & Kapros, 2015)

Principle	Description
1. Reusable functionality (structurally appropriate behavior)	The Adaptative Information system (AIS) is a system that can support any conceptual model. Domain-dependent code and structures are avoided.
2. Known categories of data (semantically appropriate behavior)	Each entity type is associated with one or more predefined generic categories. Category-specific functionality is invoked at run time for each entity type.
3. Adaptive data management (schema evolution)	The AIS can store and reconcile data with multiple definitions for each entity type (i.e., multiple conceptual models), allowing the end user to make sense of the data.
4. Schema enforcement (domain and referential integrity)	Each item of stored data conforms to a particular entity type definition, which was enforced at the time of data entry (or last edit).
5. Entity identification (entity integrity)	The stored data relating to each entity are uniquely identified in a way which is invariant with respect to a schema change
6. Labeling (data management)	The stored data relating to each entity are labeled such that the applicable conceptual models can be determined

Table 8.2. Model hierarchy of OMG

Layer	Description
M3MOF	Defines a general formal language for specifying meta-models. Example: OMG's Meta-Object Facility (MOF)
M2Meta-model	Defines a language for specifying models. Example: meta-model CI, openEHR
M1User models	Defines a language for describing semantic domains. Example: a model for medical knowledge, a model for patient's health
M0Instance models	Contains runtime instances of the data in the models

8.3.1.2. CI and transparency

It will be easy for IT departments or organizations to connect or remove functionality to/from CI applications because of CI's open meta-model. We refer to connectivity as seen by Byrd and Turner (Byrd & Turner, 2000) and Chanopas(Chanopas, Krairit, & Ba Khang, 2006). Connectivity will improve accessibility and use of other applications and thus advance transparency. Transparency makes the system behave in a seamlessly integrated way because users do not expect the system to reinvent or reassemble components every time a different functionality is required; cf. Star (Star, 1999). Tafti (Tafti, et al., 2013), Pavlou and El Savy(Pavlou & El Sawy, 2006) describe cross-functional integration as wide accessibility and broad use of IT resources. We expect that the meta-model of CI strengthens the connectivity of applications and thus supports transparency. Based on the above we define the following:

Proposition 3. *Because conceptual models and application structures have been separated in the IS, the accessibility of data structures and data is greatly enhanced.*

8.3.1.3. CI and Standardization of data as bases for interoperability

Standardization of the IT infrastructure can also aid in connecting different applications to one another. Hanseth and others (Hanseth, Monteiro, & Hatling, 1996) describe the tension between flexibility and standardization. On the one hand, standardization can increase flexibility, because of openness for further changes. However, on the other hand, standardization decreases "interpretive flexibility" by freezing semantics. Because semantics are essential in healthcare, information needs have shifted from the exchange of technical data to the exchange of meaning, the characteristic of semantic interoperability (Garde, Knaup, Hovenga, & Heard, 2007). In openEHR, contrary to Hanseth's conclusion, working with flexible conceptual models opens up interpretive flexibility. Semantic interoperability is defined as enabling the users to work with and exchange meaningful information. In healthcare organizations IS will enable users to communicate about care processes in medical terminology (Garde, et al., 2007). CI decouples domain knowledge from application logic and thereby supports working with standards such as openEHR, when concepts can be extended by medical professionals with explicit attributes. We therefore define the following:

Proposition 4. *When conceptual models and application structures have been separated (CI), a meta-model of medical knowledge models leads to interoperability in general, and semantic interoperability in particular.*

8.3.1.4. CI and scalability

We presuppose that the upscaling of systems to service a growing number of users concerns primarily the technical attributes of IT infrastructure (Vaquero, Rodero-Merino, & Buyya, 2011). Scalability can be improved with technical means, such as the distribution of network traffic. The application traffic across a cluster of servers can be optimized by, for instance, load balancers, virtualization, pooling of applications and application of containers. We do not expect that a meta-model for conceptual models will affect this dimension. Therefore, scalability is mainly independent of the flexibility of concepts and models in IS (Bass, Clements, & Kazman, 2012). Following this line of reasoning we define:

Proposition 5. *When conceptual models and application structure have been separated in the IT infrastructure, we do not expect effects on performance or scaling, because the technical infrastructure can be optimized independently of CI.*

8.4. Mixed-methods research in a multiple case study

8.4.1. Multiple case study

In this study the objective is to clarify the influence of CI, an underlying software design, on IT flexibility. Because CI is not directly visible to the IT professionals, it is fitting to carry out case studies. A case study will offer the opportunity to study the phenomenon of CI in-depth and collect information about the IT flexibility of an organization from different angles (Yin, 2002).

To further study the influence of CI, we compare organizations that have implemented CI to organizations that have not implemented CI in the IT infrastructure.

We had the opportunity to study CI via openEHR in the Netherlands because a group of mental healthcare organizations had started to apply modules that are based on openEHR since 2016-2017. The cases consist of the largest organizations in mental healthcare in the Netherlands. Nine out of ten are organizations with 500-800 full-time equivalent (fte) employees, one has a smaller number of employees. The application of openEHR is more common in, for instance, Norway, the United Kingdom, Australia and Russia (openEHR Foundation, 2020). We have divided the organizations into two groups:

- The "openEHR organizations" have 2 or more openEHR modules implemented and operational in the IT infrastructure and describe their organization in the interview as an openEHR organization (Cases C1 to C5)
- The "other organizations" have no or one module installed, and explicitly express, that they do *not* perceive the organization as an openEHR organization (Cases C6-C10).

8.4.1.1. Quantitative data and interviews

Lee (Lee, 1989) describes four problems that have to be solved in case study research. In this section we explain how these four problems have been addressed. First, the observations have to be carried out in a controlled way. Our collected quantitative data consists of scores of the IT architect or IT functional manager on a questionnaire. The questionnaire is based on an empirically validated survey of Mikalef et al. (Mikalef, et al., 2016; Rogier Van de Wetering, Mikalef, et al., 2018). Next to the quantitative data we performed interviews with IT professionals in the organization. All 17 hours of interviews were transcribed, then 885 text fragments were distinguished and summarized in English. The English statements that resulted were treated as qualitative data. We have made references to the observable characteristics of CI, see par. 3.1 in the interviews.

8.4.1.2. Deductions, replicability and generalizability

Lee argues (Lee, 1989) that deductions have to be made in a controlled way. We rely on a mixed-methods design, see the next paragraph. For replicability we compared five cases that all have implemented CI and explored similar patterns or effects. Finally, we allow for generalizability by describing how we have found the mechanisms of CI influencing IT flexibility and compare multiple cases with CI to multiple cases without CI. The sample of the five organizations that have implemented CI and the five organizations that have not implemented CI is not large enough to draw conclusions exclusively based on scores, therefore we integrate the quantitative and qualitative data.

8.4.2. Mixed-methods research

For deductions and interpretations of the data quantitative and qualitative data have been synthesized based on propositions. Each confirmed proposition (see 3.1) will strengthen the relation be tween CI and IT flexibility. Mixed-methods research is defined by Creswell and Creswell (Creswell & Creswell, 2005) as "… a research design or method ology for collecting, analyzing and mixing both quantitative and qualitative data in a single study or series of studies in order to better understand research problems". Their method consists of collecting both quantitative and qualitative data and sequentially integrating quantitative and qualitative methods. We have collected quantitative data with a questionnaire asking respondents to rate items. The data in this study were collected mostly simultaneously. In five cases one person (an IT architect) in the organization filled in the questionnaire during the interview. In three cases two persons (an IT architect and a functional management expert) did so during a combined interview. In two cases, there were two separate interviews with the IT professionals. The semi-structured interview was ordered by reading aloud each item text of the questionnaire by the interviewer. Then the interviewer asked the IT professionals to reason aloud about rating the item. Eventually the item was scored by the IT professional. At the end of the interview the interviewer specifically asked if there existed a relation to openEHR and/or observable CI characteristics, if this had not been mentioned before.

Our approach is very similar to the mixed-methods approach of Dennis in exploring the adoption of use of group support systems (Dennis & Garfield, 2003). Dennis applies qualitative data analysis and adds a quantitative analysis to enhance understanding. In that process, he uses statistical tests on small samples and combines quantitative and qualitative data, to strengthen the results. He adds: "This mixed method of utilizing quantitative survey data with a small N to complement qualitative data has become accepted in IS research." (Dennis & Garfield, 2003).

In our study, quantitative data have been collected for assessments of IT flexibility in the organizations. We have applied quantitative data analysis by applying the Mann-Whitney U test for small samples to compare means on IT flexibility of the two groups of organizations (Nachar, 2008). The test results indicate whether organizations' scores differ. Also, visual analysis has been performed on the mean scores of items for organizations.

This study collects qualitative data to enrich the interpretation of questionnaire scores. We have used a qualitative analysis based on content analysis of interviews (Mayring, 2004; Schmidt, 2004). Qualitative data were categorized according to items and then to propositions during the analysis process. An independent, external researcher separately reviewed the ordering of texts with propositions. Fetters and others (Fetters, Curry, & Creswell, 2013) extend on methods to collect and integrate quantitative and qualitative data. We have applied "merging" for integrating data because we have brought quantitative data and qualitative data on the same items together for analyzing. At the reporting level, we have weaved data through narrative.

8.5. Results

In this section, we first present the total mean scores of two groups of organizations on IT flexibility, "openEHR organizations" C1 – C5 and the not openEHR organizations (referred to as "other organizations") C6-C10. Then, in Fig. 8.2., we compare mean scores on all items. For a complete overview of the used measures, see the Appendix. Then we will integrate these differences in a narrative with explanatory remarks from interviews.

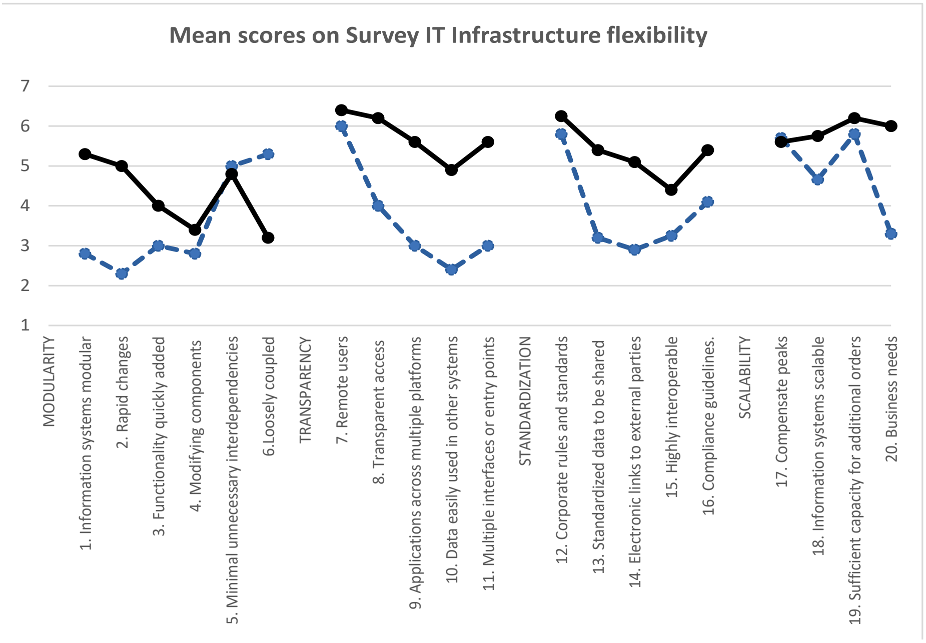


Fig. 8.2. Mean scores of openEHR organizations and other organizations on the questionnaire of IT infrastructure flexibility

8.5.1. Difference in total mean scores

The total mean score on IT flexibility of openEHR organizations is 5.2 on a scale of 1-7, and the total mean score of other organizations is 3.9. The Mann-Whitney U test in the package SPSS has been executed on total mean scores. The Mann-Whitney U test for small samples is a nonparametric test. The null hypothesis states that the distributions of mean scores of both populations are equal (Nachar, 2008). The hypothesis that the distribution of IT flexibility scores is the same across both groups is rejected with a significance level of 0.05.

The null hypotheses that the distribution of scores for dimensions of IT flexibility of both populations are equal can be rejected for transparency and standardization with significant level of 0.05. The null hypotheses that distributions of scores of organizations are equal for modularity and scalability have been retained.

8.5.2. Overall differences between groups on IT infrastructure flexibility

The visual inspection of data in Fig. 8.2. unfolds that the mean scores of the group of openEHR organizations are higher for the various items of IT infrastructure flexibility. Observation of data in Fig. 8.2. shows remarkable differences for groups of organizations for modularity. First openEHR organizations score seemingly higher than the other organizations on items 1-3. These items concern the adaptivity of IS. For items 4 and 5 (i.e. interdependencies in the IT infrastructure) the two groups' mean scores are about the same. On the item regarding the loose coupling of systems, item 6, the other organizations score higher. For items on transparency and standardization the openEHR organizations score higher overall. For scalability items no clear pattern emerges.

8.5.3. P1: Decoupling the conceptual models from application logic and modularity

In this section the quantitative data has been extended with interview data. We have analyzed the data belonging to every item in the survey. We searched for reasons for the score that the organization selects. The reasons did not have to be interpreted, because they were explicitly worded by the interviewees. We have analyzed interview statements to establish if there exists a relation to openEHR. The relation exists if the interviewees mention openEHR in their reasons. Fig. 8.2. shows that openEHR organizations score higher on items 1-3 related to adapting IS and thus on the modularity of IS. Item 1 explicitly asks for the modularity of IS. In Table 8.3. comments and responses of interviewees of organizations have been summarized. Four out of five openEHR organizations (C1, C2, C3, C5) express the ease with which applications of openEHR can be changed. The following two excerpts from C1 and C5 clarify this view:

Table 8.3. Adaptivity and IT infrastructure flexibility

P1: Existence of AIS increases IT infrastructure flexibility	
openEHR and flexibility	**other organizations: how to realize flexibility**
C1 The openEHR systems are easy to change, and this affects the infrastructure where these modules are positioned.	C6 This organization needs functionality in integrated ways to accommodate the care professional. The IT department wants to deliver independent components, but the care professional needs information systems to exchange information and operate together in an integrated EHR.
C2 Changes in the information systems are possible. It is possible to change the application infrastructure.	C7 This organization confirms that modularity is necessary for flexible systems and that the mental healthcare sector needs flexible systems. Now the systems have evolved into inflexible systems.
C3 After a history of collaboration with different vendors in the past, the organization experienced that the vendor of openEHR could execute changes in systems easily.	C8 The organization is dependent upon changes made by the vendors of the applications.
C4 This IT professional thinks the IS are not easy to change, just average. He looks at the complete IT infrastructure and does not see possibilities for rapid change, especially in critical applications (not openEHR).	C9 The organization employs full time software developers to adapt the information systems to a certain extent. The organization does not have to wait for the vendor to act.
C5 The openEHR part of the infrastructure differs from the rest of the IT infrastructure. The openEHR systems can be easily and rapidly changed.	C10 The IT professional thinks that vendors have difficulty in changing the information systems.

“The openEHR part of the infrastructure differs from the rest of the IT infrastructure, the openEHR systems can be easily and rapidly changed. The information layer of the openEHR modules is more difficult to change, but it can be done. We see a world of difference between the openEHR part and the other part in our systems.”

“The vendor of openEHR offers the functionality that we do not have in our own EHR (Electronic Patient Dossier). Our experience with openEHR software is that changes can be made, that are difficult in other software. We see that easy change is possible in new software.”

One of the organizations (C4) implemented several modules of openEHR but did not replace critical IT applications. As a result, difficulties were experienced when connecting different types of IS that is openEHR and other IS, to each other. Summarizing, four out of five confirm that CI characteristics affect the flexibility of the IT infrastructure in a positive way.

When we look at the five other organizations in Table 8.3., all describe that change in the IS is challenging to realize. In organization C10 reuse has not been recognized:

> “Reuse of functionality with different conceptual models probably has not been implemented in our EHR. When I look at the application of our EHR, I can conclude that the system is not very flexible, that it is difficult to change. I base this conclusion on discussions in the User Group of mental health organizations. There are not many alternatives for our EHR.”

Regarding the same topic, the informant of C8 adds:

> “No, we cannot realize rapid changes. Changes go slow, promises are not being followed up. On the other hand, we do have the possibility to develop our own forms in applications. The creation of forms is the responsibility of functional management.”

A third organization (C9) employs full-time software developers and executes the changes itself and adds customized/self-made functionality that has to be maintained by the organization.

The interviews confirm that openEHR applications are easier to adapt than other IS. The other organizations complain about the lack of speed for adapting applications, except for the applications that can be reprogrammed by internal developers (C9).

8.5.4. P2: Reuse of functionality and modularity.

In Table 8.4. an overview of statements of ten organizations on P2 has been represented. In openEHR and CI reuse of functionality with different conceptual models is feasible, cf. 2.5. Two openEHR organizations (C3, C5) express a positive effect of the reuse of functionality on modularity. One (C1) openEHR organization attributes a positive effect on modularity to Data reuse and not to Reuse of functionality. The IT architect of C1 states:

> “The openEHR part of the infrastructure has been structured modularly. openEHR has technical characteristics to integrate with the software of other vendors, but these vendors are not making an effort.”

Contrary to expectations, C2 and C3 score low on item 6, they do not experience loose coupling of IS and IT infrastructure. These organizations have implemented a major number of openEHR modules. C3 implemented 10 out of 15 modules, while the IT architect of C2 says that practically all have been implemented. The organizations C2 and C4 point to the effect of connecting applications, specifically connecting openEHR and not openEHR applications. IT architect of C4 gives an explanation:

> “… the problem arises when you have to migrate the data. We see that the interconnections have to be changed. Yes, changing components does affect the IT infrastructure.”

Two of the other organizations see reusable functionality incidentally in applications. Also, other organizations emphasize the benefits of modularity but have not achieved modularity unless it is implemented by the vendors.
The functional manager from C9 expresses a remarkable perspective:

"In our organization you can look at modularity from two points of view. When I look at modularity from the IT infrastructure, then the systems are highly modular. When I look at modularity from the point of view of applications, the systems are not modular."

In summary, the openEHR organizations do not score lower than other organizations on modularity items 4-5, but they score lower on item 6, regarding loosely coupledness of systems. The openEHR organizations express difficulties in connecting applications, although they do not attribute the difficulties to openEHR.

Table 8.4. Reuse and modularity

P2: Reuse of functionality increases Modularity of IT infrastructure	
openEHR and modularity	**other organizations: how to realize modularity**
C1 This organization focuses on data reuse instead of reuse of functionality. This affects the IT infrastructure because data is easily exchanged.	C6 This organization is requiring a modular structure but needs the functionality of the integrated EHR system.
C2 Building blocks for functionality have been perceived in the IT infrastructure.	C7 In this organization the required modularity is not present. The IT infrastructure evolved: first, the EHR is selected and its functionality is evaluated. Then for missing functionality other applications have been added to complete the IT infrastructure.
C3 The reuse of functionality in the software of openEHR is recognized. The reusable functionality is used by half of the software.	C8 In this organization the required modularity is not present.
C4 The software of openEHR can process different conceptual models.	C9 This organization sees modularity as useful. Reuse of functionality is seen incidentally in a data warehouse and in architecture tool.
C5 Reusable functionality is an integral part of openEHR software. However, the organization does not apply it fully at this moment.	C10 In this organization, the IS is modular. Reuse is possible by switching modules off and on, but there is a need for more reusable modules.

Table 8.5. Accessibility of data

P3: Accessibility of data structures and data is greatly enhanced with CI	
openEHR and accessibility	**other organizations: how to realize accessibility**
C1 ICT has been integrated completely in the care processes. The organization is now in a change process from unstructured data to a Best of Breed architecture with openEHR.	C6 In this organization an integrated solution is needed and found in the EHR.
C2 There is transparent access to all platforms and applications. If users have rights, then access is possible	C7 Remote users can seamlessly access centralized data and processes. However, data exchange is possible, but not easy.
C3 Accessibility of data has been the main reason to start with openEHR.	C8 The organization describes situations in which data is hard to access.
C4 The new EHR (openEHR) does offer access to other applications through an SSO. It is web-based.	C9 In this organization the data of one system can easily be used in other systems. This organization can manage more than one data model in the database of the EHR, because of self-made software.
C5 The organization applies openEHR, because it can integrate easily between the ETL and our own EHR for extracting data for reuse elsewhere.	C10 This organization has plans for improving data exchange. No, this is not easy.

8.5.5. P3: Accessibility of data structures and data is greatly enhanced with CI

The mean scores of the openEHR organizations on transparency, items 7-11, are higher than the mean scores of the other organizations. See Appendix. In Table 8.5. an overview of statements of ten organizations on P3 has been represented.

A crucial item for transparency is item 10: Data of one system can easily be used in other systems. The mean of the openEHR organizations is 4.9 (7 points scale), the mean of the other organizations is 2.4 (7 points scale). The openEHR organizations score considerably higher. Four out of five openEHR organizations see the ease of Data access as a characteristic of openEHR modules (C1, C3, C4, C5), but three (C2, C3, C4) also experience difficulties in integrating with other IS.

Also, in C5, extra flexibility of openEHR is noted (CI Principle 3):

"In a standard EHR there exists one process for all care workers and it is individualistic. We more often work in groups. In openEHR software you can serve different groups with the same information."

The other organizations attribute accessibility to one of the following: CITRIX, Remote desktop or they depend on the integrated EHR. But all five other organizations express difficulties in exchanging and using data of other systems and mention infrastructure solutions and technical standards for sharing data. For example, as the IT architect from C10 expresses:

"Data of one system cannot easily be used in other systems. This is not easy, maybe if you use the raw database data, then you can access the information in other systems."

For using data in other systems, the interviewees in C8 observe that:

"It is complicated to work with the standard EHR when you are on the road with a client. You do not want to go through a login on CITRIX with a heavy laptop."

Summarizing we would say, transparency has been scored higher in openEHR organizations and evaluations in interviews give explanations why.

8.5.6. P4: The (standard) meta-model of conceptual models in CI will lead to interoperability in general and semantic interoperability in particular

This section compares the scores on items 12-16 (i.e. standardization) for openEHR organizations and other organizations. The openEHR organizations score high on all items, the other organizations only on item 12. Looking at semantic interoperability a relevant item is item 13: the organization has identified and standardized data to be shared across systems and business units. IT professionals in almost all organizations comment on this item. The openEHR organizations have a mean score of 5.4 (scale of 7) compared to a 3.9 mean score (scale of 7) of other organizations.

Table 8.6. Semantic interoperability

P4: Meta model of conceptual models in CI will lead to semantic interoperability	
openEHR and semantic operability	**other organizations: how to realize semantic operability**
C1 In the old EHR the data has not been structured. The main part consists of text fields (free text). Now part of the data has been structured in openEHR.	C6 The EHR offers an integrated solution for this organization. More than the software of openEHR could offer.
C2 The organization has identified and standardized data to be shared across systems and business units. But this IT professional thinks openEHR has made a start and working with healthcare data can be further improved.	C7 This organization has not identified and standardized data to be shared across systems and business units. There is one centralized EHR. It cannot serve a diversity of users as they need.
C3 The data models can be redesigned for translation and reuse, but the organization cannot exchange data this way.	C8 New regulations in the Netherlands demand that information has to be exchanged in a new standard for healthcare information, Care Information Building blocks (in Dutch ZIBs). This will be difficult for the vendors.
C4 The organization has identified and standardized data to be shared across systems and business units. openEHR is not fully applied here. Our systems were selected in order to incorporate external links to external parties. This was one of the reasons for choosing openEHR software.	C9 The organization has identified and standardized data to be shared across systems and business units. Remote users can seamlessly access the system. This organization develops part of its software.
C5 The organization has a standard for data to be shared across systems and business units.	C10 The organization works on sharing data between different business units.

The openEHR organizations declare that they are working on semantic standards. All name openEHR in this regard. Organizations refer to the new standard for information as in the Netherlands (ZIB), Care Information Building Blocks (Nictiz, 2017).

One organization, the one that has implemented openEHR fully, thinks that real semantic interoperability will still take a long time (C2). He refers to openEHR as follows:

> "In the Netherlands the ideas for ZIBs aim at the same goals (as openEHR) to provide building blocks to assemble different combinations of functionality. Too bad they have not chosen openEHR for this purpose."

In the interview this informant reiterates that the openEHR standard is working but that the organization expects more now; it does not progress far enough. He adds:

> "In my opinion the vendor should offer more customization in the integration. Smart flow functionality. We want to work more rule-based. We are waiting for a rule engine, that has been planned, but has not been realized yet. We can add simple forms ourselves, but it is not yet enough."

In the interviews the informants from the other organizations describe their efforts and share their problems in exchanging data between business units. One organization bases all exchange of data on the existing EHR (C6) and awaits future developments that will involve the mental healthcare sector as a network. One Other organization sees the EHR as determining and restricting the data exchange (C7), another (C9) mentions the proprietary data model to which self-made tables are added, one (C8) calls legacy software as a restricting factor. This organization expects difficulties on this item, as expressed by its representative:

> "The EHR vendors have an obligation to unlock the data in their systems in the Dutch ZIBs standard. It will be very difficult for our EHR vendor to implement the ZIB standard because of the outdated architecture. I do not recognize highly interoperable systems."

Lastly, C10 perceives an organizational gap between the functional IT department and the technical IT department, which is hard to bridge.

In Table 8.6. an overview of statements of ten organizations on P4 has been represented.

Summarizing, we observe that all ten organizations are aware of semantic operability in the context of implementing ZIBs, but the other organizations express more difficulties in the process.

8.5.7. P5: Effects on scalability are not present for CI

We find differences in scalability on the mean scores of openEHR organizations when compared with the mean scores of other organizations on items 17-20. The item scores do not have higher scores for one or the other group of organizations overall. The factors that organizations mention when referring to scalability, do not involve the openEHR software. All ten organizations (openEHR and other organizations) mention different aspects or factors that affect scalability such as: a direct dialogue between functional management and technical management, virtualization of all applications, databases, and services and hardware limits. Adaptations in software and hardware are also possible as long as those are paid for. In summary, all ten organizations mention different factors than openEHR for influencing scalability.

8.6. Discussion, conclusion and limitations

8.6.1. Discussion and conclusion

For this multiple case study, results indicate that a difference is found in IT flexibility between openEHR organizations and other organizations based on a difference in total mean score and differences in mean scores on the dimensions of IT flexibility questionnaire. The statistical results demonstrate significant differences of scores of groups of organizations. The validity of the measurement relies on the validity of the questionnaire. The questionnaire contains the same items as a validated survey of Mikalef and Van de Wetering (Mikalef, et al., 2016), therefore we presume that IT flexibility scores represent IT flexibility in the organizations. If the underlying CI design is a factor in improving IT flexibility then we expect the propositions P1 to P4 to show the influence of CI and P5 to be independent of CI. In Table 8.7. the results of quantitative and qualitative data have been summarized.

Overall, the results confirm expectations that openEHR affects the dimensions of IT flexibility when performing a visual inspection. For evaluating the influence of CI, mean scores have been integrated with the detailed comments of the IT architectures and functional management. We view the results on the propositions P1-P5 that directly formulate expected effects on IT flexibility dimensions, if CI is involved. We do find confirmation for P1, P3, P4, and P5.

Table 8.7. Overview Propositions, combined results

OpenEHR organizations	other organizations	Confirm*
P1 openEHR organizations describe higher IS flexibility and see that it affects whole infra	Express that change in IS is difficult, and it affects whole infra	V
P2 Extra opportunities for Reuse of functionality are present, but no general effects are seen on the modularity of the IT infrastructure	Modularity and Reuse of functionality are responsibility of vendors. Organizations observe a modular structure of applications only on a high level	X
P3 In general, connectivity and sharing data are realized. These are related to openEHR and affect transparency	Integration is realized with different technical means; the focus is on technical exchange of data	V
P4 Business unit collaboration is in progress, openEHR plays a role	Business unit collaboration described as difficult because of different factors relating to existing systems	V
P5 Scalability independent of openEHR	Scalability affected by various technical factors	V

* V = confirmed, X = not confirmed

For P2, we find that CI Principle 2 has been detected by the openEHR organizations in openEHR software. Moreover, the other organizations have not identified CI Principle 2 in IS. This difference in observed CI principles confirms our assumption that CI has been applied in openEHR but not in IS proprietary software. P2 is not confirmed by the data. The openEHR organizations observe more modularity in IS, item 1, as expected, but do not experience loosely coupled components in the IT infrastructure, item 6. The qualitative data show that some openEHR organizations realize modularity with openEHR, while other openEHR organizations do not. More research is needed here.

Despite its contributions, the present study is constrained by several limitations that future research should seek to address. First, we have not performed direct research on CI; we relied on the openEHR standard (the 'proxy'). However, all underlying design principles of CI have been detected in openEHR software. The explanations of the IT professionals in the organizations confirm the role of openEHR. We infer the influence of CI from their remarks. Secondly, the implementation of openEHR depends on one vendor in the Netherlands, which is not the case in other countries (openEHR Foundation, 2020).

8.6.2. Implications for practice and theory

Hellberg describes three strategies for healthcare organizations to enhance IT (Hellberg & Johansson, 2017). These are: technological strategy, a governance strategy, and a political/organizational strategy. The technological strategy focuses on building health portals, as the foundation of organizational IT applications. IT flexibility is indispensable for extending the infrastructure with eHealth applications. The governance strategy increases attention for management to the self-determination of the individual in improving public health, thereby focusing on the internal business processes and organizational processes. The third strategy positions ICT as a means for empowering individuals to take responsibility for their own health and thereby increasing equity for different population groups in accessing healthcare.

Our research explores the software design principles for increasing IT flexibility in IS concerning medical and healthcare terminology. In literature about openEHR and medical terminology we found that flexible domain knowledge is essential for healthcare professionals. The flexibility of conceptual models is mostly neglected in the extant literature on IT flexibility. Our contribution indicates that the way the domain model has been implemented, plays a pronounced role in the flexibility of organization-wide IT infrastructure. For theory building we advice that the mechanisms that cause the effects of CI on IT flexibility, such as the scope of the meta-model of conceptual models, should be further examined.

This research can be positioned in the technological strategy, but its implications are not independent of healthcare policy in the organization. Building IS that contain a perfect and unchangeable (hard) copy of the terminology at a certain moment in time will impede further evolution of healthcare policy. IT departments in healthcare organizations encounter dependencies in traditional EHR software, when conceptual models are integrated and encapsulated in closed software. The inflexibility of conceptual models obstructs further evolution of data models. The encapsulation leads to difficulties when meaningful data is necessary for healthcare processes. Healthcare organizations need to decide on their IT policy, specifically how much openness about “black box software” they require from vendors in order to realize IT infrastructure flexibility in organizations.

Ethical approval Not required

Data statement The quantitative data and interviews that have been studied in this article can not be made public. The representatives of the organizations have asked the authors not to publish the raw data; all interviews have been confidential. The organizations have given permission to Utrecht University and HAN university of applied sciences to transcribe and analyze the data for research and use quotes from the interviews in publications.

Funding HAN University of Applied Sciences, the Netherlands

Declaration of Competing Interest None declared

Supplementary materials Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.hlpt.2020.04.001.

Appendix

Mean scores on IT flexibility questionnaire

Tables 8.8.– 8.12.

Nonparametric Tests

Test have been executed with SPSS software package with a configuration of significance = 0.05 and confidence interval = .95. The asymptotic significances are displayed.

Table 8.8. Mean Scores on IT Infrastructure Flexibility Questionnaire

Groups of organizations	Other organizations	openEHR organizations
MODULARITY		
01. Our information systems are highly modular	2.8	5.3
02. The manner in which the components of our information systems are organized and integrated allows for rapid changes	2.3	5
03. Functionality can be quickly added to critical applications based on end-user requests	3	4
04. Exchanging or modifying single components does NOT affect our IT infrastructure	2.8	3.4
05. Organizational IT infrastructure and applications are developed on the basis of minimal unnecessary interdependencies	5	4.8
06. Organizational IT infrastructure and applications are loosely coupled	5.3	3.2
TRANSPARENCY		
07. Remote users can seamlessly access centralized data and processes	6.0	6.4
08. Our user interfaces provide transparent access to all platforms and applications	4.0	6.2
09. Software applications can be easily transported and used across multiple platforms	3.0	5.6
10. Data of one system can be easily used in other systems	2.4	4.9
11. Our firm offers multiple interfaces or entry points (e.g., web access) to external users.	3.0	5.6
STANDARDIZATION		
12. We have established corporate rules and standards for hardware and operating systems to ensure platform compatibility	5.8	6.3
13. We have identified and standardized data to be shared across systems and business units	3.2	5.4
14. Our systems are developed in order to incorporate electronic links to external parties	2.9	5.1
15. Organizational IT infrastructure and applications are highly interoperable	3.3	4.4
16. Organizational IT applications are developed based on compliance guidelines.	4.1	5.4
SCALABILITY		
17. Our IT infrastructure easily compensates peaks in transaction volumes	5.7	5.6
18. Our information systems are scalable	4.7	5.8
19. Our IT infrastructure offers sufficient capacity in order to fulfill additional orders for treatment or diagnosis	5.8	6.2
20. The performance of our IT infrastructure completely fulfills our business needs regardless of usage magnitude	3.3	6.0
TOTAL MEAN	3.9	5.2

Scores on scale 1 -7 (1 totally disagree, 7 totally agree)

Table 8.9. Mean Scores for Organizations on IT Infrastructure Flexibility Questionnaire

ID – random	IT Flexibility score	Organization-type
1	3,52	Other
2	4,65	Other
3	4,44	Other
4	3,40	Other
5	3,56	Other
6	6,70	openEHR
7	4,45	openEHR
8	5,40	openEHR
9	4,50	openEHR
10	5,03	openEHR

Table 8.10. Mean Scores for Organizations
on Dimensions of IT Infrastructure Flexibility Questionnaire

ID – random	MOD *	TRANS*	STAND*	SCAL*	Organization
1	3,58	2,50	3,70	4,20	Other
2	3,83	4,80	4,00	6,50	Other
3	3,40	4,25	4,80	5,50	Other
4	3,00	3,20	4,10	3,38	Other
5	3,70	3,25	2,50	4,75	Other
6	6,50	7,00	6,40	7,00	openEHR
7	3,67	5,60	4,40	4,25	openEHR
8	4,67	5,20	6,20	5,75	openEHR
9	2,67	5,00	4,80	6,25	openEHR
10	3,90	5,90	4,38	6,33	openEHR

*) Dimensions are: MOD = Modularity, TRANS = Transparency,
STAND = Standardization, SCAL = Scalability

Table 8.11. Hypothesis Test Summary Means

Null hypothesis	Sig.	Test	Decision
1 The distribution of IT Flexibility Scores is the same across categories of Groups of Organizations.	Independent-Samples Mann Whitney U Test	.032[1]	Reject the null hypothesis.

Table 8.12. Hypothesis Test Summary – Dimension Means

Null hypothesis	Test	Sig.	Decision
1 The distribution of MOD* is the same across categories of Groups of Organizations**	Independent-Samples Mann Whitney U Test	.310[1]	Retain the null hypothesis.
2 The distribution of TRANS* is the same across categories of Groups of Organizations**	Independent-Samples Mann Whitney U Test	.008[1]	Reject the null hypothesis.
3 The distribution of STAND* is the same across categories of Groups of Organizations**	Independent-Samples Mann Whitney U Test	.032[1]	Reject the null hypothesis.
4 The distribution of SCAL* is the same across categories of Groups of Organizations**	Independent-Samples Mann Whitney U Test	.222[1]	Retain the null hypothesis.

*) Dimensions are: MOD = Modularity, TRANS = Transparency, STAND = Standardization, SCAL = Scalability
**) Groups of Organizations are: openEHR organizations and other organizations
[1] Exact significance is displayed for this test.

9 Investigating the Impact of Outsourcing on IT Flexibility: The Conceptual Independence Perspective

Modern healthcare organizations try to leverage their IT infrastructures to enhance the efficiency of processes and the quality of patient services. The flexibility of the IT infrastructure is a critical factor in the process of establishing strategic and operational value. The authors examine how applied principles of Conceptual Independence (CI) in information systems (IS) influence the flexibility of IT infrastructures. Furthermore, it is presumed that IT outsourcing plays a role in IT flexibility. The second question asks whether IT outsourcing configurations change when CI has been applied or not. Quantitative and qualitative data have been collected in 9 mental healthcare organizations. Findings – based on integration of the data with a mixed-method approach - suggest that the healthcare organizations that apply the principles of CI are better equipped to adapt their IT infrastructure to changing demands, requests and needs. Likewise, results suggest that they have changed the government of IT outsourcing thereby increasing IT flexibility even further.

9.1. Introduction

The healthcare sector in the Netherlands has to meet a diversity of new requirements. First, similar to the situation in other countries, demographic factors influence healthcare processes. Secondly, new government regulations are directed towards empowering patients to take control of their patient data.

Therefore, care processes and supporting IT must be reorganized.

Furthermore, new technologies, such as mobile devices and apps, internet of things, cloud services and smart medical devices have the potential to realize gains in quality of care. Data from different devices need a connection to data in large electronic health records (EHR) systems. At the same time IT departments struggle with the pressure to decrease costs. The challenges have forced IT departments to outsource the software development function. Therefore, they must now rely on standard packages of Commercial-Off-The Shelf (COTS) software systems.

Submitted for publication:

Tarenskeen, D., Van de Wetering, R., Bakker, R., & Brinkkemper, S. (2020). *Investigating the Impact of Outsourcing on IT Flexibility: The Conceptual Independence Perspective*.

When studying the research literature on flexibility of IT infrastructure, dynamic capabilities have been distinguished in firms that were able to adapt to new circumstances (Eisenhardt & Martin, 2000; Teece, Pisano, & Shuen, 1997). These capabilities involve among others sensing and seizing opportunities in the environment and responding to changes with reconfiguration of business processes. Reconfiguration of business processes needs reconfiguration of IT infrastructure. Therefore, IT flexibility appears a critical enabler for dynamic capabilities according to Mikalef et al. (Mikalef, Pateli, & Van de Wetering, 2016) and needs to be aligned with dynamic capabilities in firms (Van de Wetering, Mikalef, & Pateli, 2018). Conceptual independence (CI) (McGinnes & Kapros, 2015) can improve flexibility in IS. In CI conceptual models in software have been separated from the application logic, thereby creating loosely coupled components in software. The claim is made by McGinnes and Kapros that implementing CI leads to better adaptable IS (McGinnes & Kapros, 2015).

The authors demonstrate in this paper that CI can be detected as an underlying design principle in openEHR software that is responsible for the flexibility of IS. In healthcare organizations the software development function has often been outsourced. This leads to the question of how collaboration with IS suppliers is functioning in the context of CI. This research investigates differences in IT outsourcing (ITO) practices related to CI. We use the term supplier instead of vendor to emphasize that the supplier is the owner and developer of software applications.

Hence we formulate the research question as: *"What are the direct effects of the combination of IT Outsourcing and Conceptual Independence on IT flexibility in healthcare organizations?"*

A comparative multi case study has been performed by interviewing five organizations that have implemented the CI design and four organizations that have not. We have applied a mixed-method approach and synthesized quantitative and qualitative data to investigate the effects of the combination. The qualitative data have priority over quantitative data in this study.

The openEHR standard provides a meta-standard for conceptual models in the medical domain. It is argued here that openEHR is exemplary for CI. As is specified on the website of the community of openEHR, "'openEHR' is the name of a technology for e-health, consisting of open specifications, clinical models and software that can be used to create standards and build information and interoperability solutions for healthcare." (openEHR Foundation, 2020d). The standard of openEHR is not a standard for medical concepts but a specification, a meta-standard, in which medical concepts can be modeled. In this characteristic the similarity between openEHR and CI can be observed. The software implementation of openEHR applies archetypes (Beale, 2002; Beale & Heard, 2007), flexible models dedicated to healthcare purposes. The openEHR standard has been applied and implemented in large-scale healthcare projects internationally (Ulriksen, Pedersen, & Ellingsen, 2017) with the intention to give medical professionals the tools, to model their knowledge independent of IT applications. We have researched IT flexibility in practice and find that new software based on the openEHR standard shows promise in opening rigid structures in software. Since we began studying the openEHR standard the number of countries applying the standard increased, now including among others Norway, Sweden, UK, China, Germany, USA, Australia, Slovenia and the Netherlands (openEHR Foundation, 2020a, 2020b, 2020c). In Norway a pilot initiated for registration and monitoring of COVID-19 patients in hospitals, has recently been deployed and is in use. In a mini symposium of Stichting openEHR Nederland (openEHR Foundation in the Netherlands), Naess reports (Næss, 2020) on a diagnosis and treatment decision support system based on openEHR models that were reviewed internationally. The software could be implemented within 11 days because the models have been added to existing openEHR implementations in hospitals. He noted that six hospitals have ordered the openHR Covid-19 software.

We focus on the effects of openEHR and CI on IT flexibility in research question 1:

"Do the organizations that have implemented Conceptual Independence observe an effect on IT flexibility resulting from the openEHR software?"

We first have to determine which organizations have observed and applied CI principles.

Next, in research question 2 the relation between ITO and IT flexibility is explored. Since organizations apply many IT applications from different suppliers, the need for integration of systems arises. The strategy of the organizations needs to include a vision on how integration of systems will be arranged. According to Teece (Teece, 2007) when organizations outsource resources and capabilities to suppliers to realize economies of scale, they run the risk to hand over control over the dynamic capability of seizing and refactoring to the IS suppliers. Teece sees "the important capability to capture value from innovation" as "the ability of the innovating enterprise to identify and control the 'bottleneck assets' or 'choke points' in the value chain." (Teece, 2007). This capability is critical for strategic management.

Research question 2 explores the role of ITO and CI:

"Do organizations that have implemented Conceptual Independence differ from the other organizations by ITO capability factors or ITO governance factors?"

We investigate if the application of CI has an influence on the control that the organization has over purchased software systems.

9.2. Theoretical background

9.2.1. IT Flexibility

IT flexibility is important for organizations that need to change. The IT capabilities in the organization have to support dynamic capabilities of sensing, seizing and reconfiguration of processes and systems. IT flexibility stands out as an IT capability, because "..the combined effect of the underlying dimensions of IT flexibility enable a firm to develop the IT-enabled dynamic capabilities that are necessary to cope with changing conditions", in a paper of Mikalef et al. (Mikalef, et al., 2016). The role of IT flexibility in a volatile environment has also been demonstrated by Teece (Teece, et al., 1997). Mikalef et al. (Mikalef, et al., 2016) conceptualize and operationalize IT flexibility with the following dimensions: modularity, transparency, standardization and scalability. Each of these dimensions strengthens IT flexibility.

The dimension of modularity has been defined as a structure of loosely coupled independent subsystems, that supports system engineers with adapting the system to new uses and requirements. According to Simon (Simon, 1962), the decomposability of the system in meaningful (technical) components will be a primary factor in the ease of changing the software system. Schilling (Schilling, 2000) emphasizes pressures in contexts or markets for products that will force products to evolve to modularity. She sees modularity as the separation of components in systems that will allow reconfiguration or recombination of components to form systems with new features. Mikalef et al. concentrate on "Loose coupling" as the main idea behind modularity (Mikalef, et al., 2016; Van de Wetering, Mikalef, & Pateli, 2017).

Summarizing, the dimension of modularity is seen as enabling flexibility in software systems (N. B. Duncan, 1995; Simon, 1965; Terry Anthony Byrd, 2000). The authors see a relation between CI and modularity, because separating conceptual (domain) models from functionality is the main characteristic in CI. Hence a separation of conceptual models from the application logic will make applications easier to adapt.

The dimension of transparency in IT flexibility refers to the possibility to use a diversity of IT applications in an existing IT infrastructure without confronting the user with the changeovers. Byrd and Turner (Byrd & Turner, 2000), and Chanopas (Chanopas, Krairit, & Ba Khang, 2006) explain transparency as seen from end-users. Transparency makes the system behave in a seamlessly integrated way to users, see Star (Star, 1999). Pavlou and El Savy refer to Cross-functional integration for this attribute of IT infrastructure (Pavlou & El Sawy, 2006). We define transparency as the seamless integration of data and processes, measured as the ease with which users can access internal and external platforms and applications.

Standardization, the third dimension of IT flexibility, has been defined as the degree of agreement in the organization on hardware, software and data standards.

The last dimension of IT flexibility, scalability, measures how well the IT infrastructure can be scaled up and upgraded, when adaptation is necessary due to growing demand and increasing number of users (Chanopas, et al., 2006).

9.2.2. IT Outsourcing

We take a new approach to ITO: we want to focus on the role that ITO plays regarding IT flexibility in context of software based on CI.

IT outsourcing has been extensively researched since 1990. An overview of the extensive research on ITO in the years 1990-2009 (Dibbern, Goles, Hirschheim, & Jayatilaka, 2004; Gonzalez, Gasco, & Llopis, 2006; Lacity, Khan, & Willcocks, 2009) show that the research in the first decade primarily focused on exploring the concept of outsourcing IS in case studies, field studies and surveys. Dibbern et al. (Dibbern, et al., 2004) describe the process of outsourcing as "..the use of external agents to perform one or more organizational activities, here positioned in the IS domain.". They describe the results on conceptualization of IS outsourcing, responding to questions as "why, what, which, how and outcomes" for IS outsourcing. When analyzing the data to discover patterns that would lead to criteria for successful sourcing strategies and practices, Dibbern et al. remain careful. No conclusions could be drawn in the fast-evolving research area. IS outsourcing covers a broad area with a diversity of studies from which the relation of outsourcing characteristics and outsourcing success is difficult to determine (Dibbern, et al., 2004). In healthcare organizations the IS in the IT infrastructure were most likely to be outsourced as researched by Lorence et al. (Lorence & Spink, 2004). The reported motivation for outsourcing consisted of a need to improve patient care and cost reduction. They also report a general satisfaction in healthcare organizations for the outsourcing outcomes. A negative influence of ITO has been remarked by Lacity et al. (Lacity, et al., 2009). Outsourcing too much of the IT activities in the organization can have negative results because in general, clients that outsourced more than 80% of their IT budget had relatively lower levels of success. Top management commitment and readiness of client firms for outsourcing prove to be important for success. Research after 2005 focuses more on the relationship between the outsourcer and client firms. Central themes are contractual governance, how to manage contracts and who takes the largest risks. Next we find the theme relational governance describing the softer issues in the relationship such as trust, shared norms, open communication and sharing of information (Lacity, et al., 2009). Results from studies in the period shortly before 2009 suggest that combinations of ITO decision and contractual governance are associated with higher levels of ITO success, as are combinations of contractual governance and relational governance (Lacity, et al., 2009).

9.2.3. IT Outsourcing and IT Flexibility

When viewing the relation between ITO and IT flexibility Lacity et al. (Lacity, Willcocks, & Feeny, 1996) prefers a selective approach to IT outsourcing, because firms have to maximize their control on core IT activities such as strategic planning. Duncan (N. Duncan, 1996) found that outsourcing was negatively related to data modularity, data compatibility and connectivity between systems. These are all indications for a decreasing IT infrastructure flexibility. Other studies in other industries, such as in banking, doubt if the IT outsourcing leads to increased flexibility and suggest that outsourcing would impede IT alignment and IT flexibility (Beimborn, Franke, Wagner, & Weitzel, 2006), cf. Beimborn et al. In this research we focus on ITO configurations described by Bui et al. (Bui, Leo, & Adelakun, 2019). In this article ITO combinations applied by the openEHR and non-openEHR organizations are compared.

Bui et al. extend the ideas of Cullen et al. (Cullen, Seddon, & Willcocks, 2005) who have explored the ITO domain to detect existing configurations of outsourcing approaches. They show that patterns of ITO exist (Cullen, et al., 2005). Bui et al. (Bui, et al., 2019) have distinguished a number of dimensions that can be applied for analyzing ITO characteristics of organizations. ITO apparently can lead to the "innovation through outsourcing" paradox, where cost reduction can impede strategic innovations, or can initiate or increase the number of potential innovations. They raise the idea that outsourcing can be applied exploitatively, reducing costs, or in an explorative way. In exploration the client's expertise is leveraged with the suppliers' expertise to explore new markets, create new products and invent new processes." In Table 9.1., Bui et al. (Bui, et al., 2019) describe capability factors of ITO and governance factors of ITO.

Table 9.1. Dimensions of ITO, according to Bui et al. 2019 (Bui, et al., 2019)

Attributes of ITO Configurations	
Capability Factors ITO	
Service Level Strategy	The type of IT capabilities and the degree to which these capabilities are accessed through outsourcing
Supplier Strategy	How outsourced capabilities are provisioned and coordinated among various suppliers, and the number of suppliers involved
Governance Factors ITO	
Commercial Relationship	How economic exchanges between client and suppliers are governed
Pricing Strategy	How payment to the supplier is determined
Contract Commitment	The agreed duration of the contract and the built-in adjustment mechanisms that allow adaptation to changing circumstances

Source: (Bui, et al., 2019)

When studying the impact of partnership quality on outsourcing success Lee and Kim (Lee & Kim, 1999) found that partnership quality may serve as a key predictor of outsourcing success. Partnership quality is characterized by participation, communication, information sharing and top management support. Mutual dependency proved to be a negative factor. In this article it is presumed that openEHR organizations communicate more frequently with the outsourcing firm about (business) functionality and terminology, because the suppliers of openEHR software have the capacity to adapt the conceptual models of IS to specific information needs.

9.2.4. Design Principle for Conceptual Models in Conceptual Independence

CI is described by McGinnes and Kapros (McGinnes & Kapros, 2015) as a software design in which the conceptual models are decoupled from the application logic. The separation of conceptual models leads to flexible IS that can be changed without radically reprogramming the software.

They describe six subprinciples that are sufficient to achieve the separation, see Table 9.2. We refer to the subprinciples as the CI principles. CI principle 3 refers to decoupling of conceptual models. CI principle 1 adds that the remaining software components must contain functionality that can be reused for each defined conceptual model. The conceptual models must conform to a meta-model for developing the generic software. CI principle 2 states that archetypical categories can initiate semantically appropriate behavior based on categories of data entities. An example is that categorizing entities as “Location” tells the software that the entity can be shown on a map.

For software to correctly identify the content of the conceptual models in input, a number of preconditions must exist: data entities must be uniquely identified independently of the conceptual models (CI principle 5), checks and constraints are enforced at input (CI principle 4) and all data are correctly labeled (CI principle 6). CI principles strengthen each other in IS to realize the complete design of CI. In Table 2 behavior refers to the functionality of the software program.

Table 9.2. Principles of Conceptual Independence

(CI) Principle	Description
1. Reusable functionality (leads to structurally appropriate behavior, *meaning behavior is based on structures of entities not on domains**)	The Adaptative Information system (AIS) can support any conceptual model. Domain- dependent code and structures are avoided. Useful generic functionality is invoked at run time for each entity type.
2. Known categories of data (lead to semantically- appropriate behavior, *meaning behavior can be linked to semantic categories**)	Each entity type is associated with one or more predefined generic categories. Category-specific functionality is invoked at run time for each entity type.
3. Adaptive data management (enables schema evolution, *meaning models can evolve using the same data**)	The AIS can store and reconcile data with multiple definitions for each entity type (i.e., multiple conceptual models), allowing the end- user to make sense of the data.
4. Schema enforcement (*guarantees domain and referential integrity, meaning data is attributed to entities consistently and correctly**)	Each item of stored data conforms to a particular entity type definition, which was enforced at the time of data entry (or last edit).
5. Entity identification (guarantees entity integrity, *meaning data is uniquely identifiable even if used in different models**)	The stored data relating to each entity are uniquely identified in a way which is invariant with respect to a schema change.
6. Labeling (application of data management, *meaning data is labeled to find the applicable conceptual models**)	The stored data relating to each entity are labeled such that the applicable conceptual models can be determined.

*Text in italics added by authors Source:(McGinnes & Kapros, 2015)

CI aims to solve an important issue in software development, namely, that changes in concepts in thinking and communicating in organizations are hard to implement in existing systems (McGinnes, 2011; McGinnes & Kapros, 2015). The causes of difficulties often originate in software with hardcoded expectations of terminology (Garlan, Allen, & Ockerbloom, 1995, 2009; Tarenskeen, Van de Wetering, & Bakker, 2018). In addition to incompatibility other authors refer to the problem of low-level dependencies in code (Lehman, 1996; Qiu, Li, & Su, 2013). CI provides an underlying software structure to vary and adapt conceptual models without breaking the programming code.

9.3. Main focus : conceptual independence in information systems in healthcare

IT departments in healthcare organizations monitor and evaluate their IT infrastructure continuously to improve service to the healthcare organization, thereby demonstrating

IT capabilities. The IT infrastructure continuously evolves when new applications are added and integrated into the organization- wide IT. Existing IT infrastructures build on certain conceptual models in EHR software that have been designed in the past at the time of requirements analysis. Added new applications need to conform to these conceptual models. If not, and different terminology is added to existing conceptual models, then end-users will not be able to comprehend the data in EHR software. Rector (Rector, 1999) states that difficulties of terminology in patient systems in healthcare have been underestimated and that this problem leads to specific issues in software for patient systems. He mentions ten topics where misunderstandings can arise, such as interpretations of medical specialists and observed test results. The need for conceptual modeling becomes apparent when medical specialists prefer free texts for registration of medical data (Cios & Moore, 2002) and previous input is hard to retrieve. However, whenconceptual incompatibility emerges in software systems due to a growing number of conceptual models, the exchange of data becomes difficult (McGinnes, 2011). open standard of openEHR not only fulfills the need to exchange data between machines but also aids medical professionals in interpreting the meaning of healthcare terminology.

The necessity to agree on concepts and relations between concepts becomes even more urgent in case of communication through one or more IS. Every IS must contain one or more conceptual models. These models structure data storage and data application. Data must be unambiguously ordered in data models to make data processing in machines even possible. Therefore conceptual models are painstakingly designed by care professionals in close consultation with IT developers.

In healthcare organizations suppliers of software often takes responsibility for adapting software to new requirements. They apply closed and proprietary data models. The closed data models become problematic, when one or more applications from different suppliers relate to the same or similar concepts. In these situations adaptation becomes complex.

In this article we specifically focus on how organizations manage healthcare data in EHR systems consisting of medical knowledge data and patient data. We investigate the way conceptual models have been implemented in data models and the effect on IT flexibility. Effects of the underlying software structure of CI (McGinnes & Kapros, 2015) are under scrutiny. Therefore the black boxes of IS have to be disclosed to analyze the software structure in EHR systems.

In 2014 Atalag et al. (Atalag, Yang, & Warren, 2014) describe how the standard of openEHR has aided software developers to change functionality in an open source medical application rapidly. Recently national news in Norway reported on an openEHR project (Trygstad & Helness, 2020). Changing conceptual models is critical in adapting to new requirements as is illustrated by the COVID-19 project for monitoring patients in 2020. The openEHR software has been applied to deploy the monitoring software in hospitals in Norway in 11 days (Næss, 2020). However, other real-life projects on changes in conceptual models related to IT flexibility have hardly been documented.

9.4. Research method

9.4.1. Mixed-methods research in a multiple case study

Two sets of data have been collected in the multi case study. We selected a mixed-method approach where results were collected with different means that complement each other to enhance understanding of the results (Creswell & Creswell, 2005).

9.4.2. Selection of Cases

IT architects and functional management in nine mental healthcare organizations have been interviewed. The IT professionals were employed in five organizations that have implemented CI and four organizations that were not implementing CI. The standard of openEHR has been selected as a proxy for CI. We presume that the principles of CI are one on one detectable in the openEHR software. Five organizations applied more than two modules of the openEHR software and described themselves as applying openEHR. These organizations are referred to as "openEHR organizations" in this article. Four non-openEHR organizations did not have openEHR modules at all or were testing modules and explicitly stated that they were not applying the openEHR software in the IT infrastructure. We see these organizations as "non-openEHR organizations". All nine cases have outsourced software development and maintenance of application software.

Table 9.3. Case organizations in mental healthcare

Organization	Number of interviewees	Number of employees in 2017 (full time equivalent)	Number of modules openEHR
openEHR	1	1000-1500	12
openEHR	2	1000-1500	10
openEHR	1	1000-1500	2 + 2
openEHR	1	1500-2000	15
openEHR	2	1500-2000	9
non-openEHR	1	less than 500	None
non-openEHR	2	1500-2000	1
non-openEHR	2	2000-2600	1
non-openEHR	1	2000-2600	1

9.4.3. Data collection

The quantitative data consist of scores on a questionnaire of 20 items on IT flexibility (Mikalef, et al., 2016). The questionnaire has been applied and validated in a large sample study of Mikalef et al. (Mikalef, et al., 2016). A full text of the questionnaire can be found in the Appendix table 9.7. The mean scores of organizations on the dimensions of IT flexibility and the total mean scores have been analyzed to find differences between the groups of organizations. Quantitative data have been analyzed with a statistical test for small samples, the Mann-Whitney U test (Nachar, 2008).

The qualitative data have been collected from semi-structured interviews performed in nine organizations with the IT architect or with the IT architect and IT functional management. The semi-structured interviews included the 20 questions in the questionnaire (Schmidt, 2004). After the interview on the questionnaire, explicit questions about the recognition and application of CI principles followed.

Questions about ITO were not included beforehand but all organizations talked extensively about the relation with the suppliers and their ITO strategy when explaining the background and history of their organization.

9.4.4. Analysis and synthesis

The 13 interviews accounted for 15 hours of interviews, distributed about evenly over the organizations. All interviews were transcribed. After transcribing the interviews, we split the text in text fragments that focus on one subject. Then we applied content analysis by summarizing the fragments into an English statement (Gläser & Laudel, 2013). Afterward we applied the codings in Table 9.8. in the Appendix to make a rough categorization of text fragments. Every statement related to an item in the questionnaire as explained by the respondent. The relation of statements to items in the questionnaire and the coding has been discussed with an independent researcher.

For measuring the application of principles of CI to increase IT flexibility we have taken the statements about CI principles and IT flexibility and transformed these into narratives one for each organization.

For the exploration of the role of the suppliers of software in all organizations all statements about the suppliers were selected (Mayring, 2004). Then statements were ordered in categories of Bui et al. (Bui, et al., 2019) in Table 9.1. (Burton-Jones & Lee, 2017; Paré, 2004). Since the study is explorative, results were mapped into a table and roughly interpreted by counting.

9.5. Results

9.5.1. Quantitative data: openEHR and IT Flexibility

Results of the quantitative data confirm that a difference exists between the openEHR organizations and the non-openEHR organizations. The mean score on the IT flexibility questionnaire of the openEHR organizations is 5.2 on a scale of 1-7 (7 is more flexible) and the mean score of non-openEHR organizations is 3.7. The total mean scores show that the openEHR organizations evaluate the IT infrastructure of their organization as more flexible than the non-openEHR organizations do, see Appendix.

A statistical test, the Independent-Samples Mann-Whitney U Test for small samples in the SPSS package is applied. We tested the null hypothesis that both sets of means originate from the same population of means. The test in Appendix Table 5 indicates that the null hypothesis can be rejected with $p < 0.05$.

A second Independent-Samples Mann-Whitney U Test has been performed for mean scores on dimensions of IT flexibility. The test only gives reason to reject the null hypothesis for the dimension of transparency with $p < 0.05$. In detail the openEHR organizations have a mean score on the dimension of transparency of 5.7 (scale 1-7) and the non-openEHR organizations have a mean score of 3.3.

9.5.2. Research Question 1A: Recognition and Application of CI Principles

To answer research question 1 we must first select the organizations that have implemented CI. In all interviews the researcher explicitly asked for recognition of CI principles in the current IS of the organizations.

Table 9.4. CI Principles in openEHR Software in openEHR Organizations

Cases	CO1	CO2	CO3	CO4	CO5
CI-1 Reusable functionality (structurally appropriate behavior)					
• Recognized	√	√	√	√	√
• Applied	√	√	√	Partly	√
• Reuse of data in architecture	√		√		√
CI-2 Known categories of data (semantically appropriate behavior)					
• Recognized	√	√	√	√	√
• Applied	√	√	√	Partly	√
CI-3 Adaptive data management (schema evolution)					
• Recognized		√	√	√	√
• Applied		√	√		√
CI-4 Schema enforcement (domain and referential integrity)					
• Recognized	√	√	√	√	√
• Applied	√	√	√	√	√
CI-5 Entity identification (entity integrity)					
• Recognized		√	√	√	√
• Applied		√	√	√	
CI-6 Labeling (data management)					
• Recognized		√	√		√
• Applied		√	√		√
TOTAL RECOGNIZED	3	6	6	5	6
TOTAL APPLIED	3	6	6	3	5

√ : perceived by respondents, empty cells: no statements have been found in the interviews.

For openEHR organizations the perceptions of the respondents have been noted. In Table 9.5., for every CI principle, a mark √ has been set in the row RECOGNIZED if the CI Principle has been observed in software. For openEHR the marks have all been related to openEHR software. If the organization explicitly states that the principle has been applied, then a mark is set in the row APPLIED. The numbers of marks have been totaled for each case organization.

As can be seen the openEHR organizations in general recognize the CI principles in the openEHR software. More than half of the CI-principles have been applied insofar as confirmed by the respondents.

Table 9.5. CI Principles in Other Software in Non-openEHR organizations

Cases	**CN6**	**CN7**	**CN8**	**CN9**
CI-1 Reusable functionality (structurally appropriate behavior)				
Recognized	in questionnaire program			
Applied	√			
Reuse of data in architecture				
CI-2 Known categories of data (semantically appropriate behavior)				
Recognized	in financial program and in BI - not in EHR system	no insight in applications at this level		
Applied	partly			
CI-3 Adaptive data management (schema evolution)				
Recognized			in data warehouse	
Applied			in management information systems	
CI-4 Schema enforcement (domain and referential integrity)				
Recognized	consistency checked by vendor		in applications	consistency checked by vendor
Applied	free text cannot be checked		in applications	
CI-5 Entity identification (entity integrity)				
• Recognized			partly in other systems	consistency checked by vendor
• Applied				
CI-6 Labeling (data management)				
• Recognized				
• Applied				
RECOGNIZED	3	0	2	2
APPLIED	1	0	2	0

√ : perceived by respondents, empty cells: no statements have been found in the interviews.

As seen in Table 9.6., non-openEHR organizations do recognize the principles of CI incidentally in other IT applications, but not in the current EHR system. Furthermore, they have no knowledge of the implemented conceptual models except in data warehouse systems or in business intelligence systems. CI Principle 4 is generally recognized with the explanation that the supplier is responsible for checking consistency in the database.

9.5.3. Research Question 1B: CI Principles in relation to IT Flexibility

Since in the interviews the openEHR organizations refer to CI principles (CI-1 to CI-6) often and non- openEHR organizations only partly recognize the CI principles, we can suffice with an analysis of the transcripts of openEHR organizations.

1. CASE CO1

In the interview with the IT architect and functional manager, the respondents observe that the software supplier is fast with adapting the information systems and that they themselves can easily adapt forms to their needs. They apply the medical database based on the openEHR structure as a means for exchanging data between applications. The IT architect plans to build a new IT infrastructure according to best of breed. They position the medical database in the center:

> "The medical database offers specific data for researchers at the university (psychiatrists). With openEHR, we can make the data in the medical database accessible without developing point-to-point- interconnections. You can access the data directly from the information layer. We are working towards a new architecture based on openEHR."

2. CASE CO2

This organization has implemented all modules of openEHR and applies all. The IT architect also remarks that openEHR is not the same as CI, because the meta-model of openEHR is limited to the medical domain where in contrast CI supports a more generic meta-model. The organization finds that openEHR improves flexibility, because functionality can be partly added by the organization itself.

The principle CI-3 is observed in openEHR, but the IT architect thinks that changing conceptual models is not always the solution, because it leads to other difficulties regarding migration of data.

Modularity has been improved by an extra separation in the IT infrastructure, but modularity cannot be achieved easily.

> "Our organization has separated the care registration and the billing system from the medical data. This is different from other organizations, where both systems have been integrated. This enables us toreplace one system or the other, but it is not a simple process."

He adds that exchanging data for working with workflows can be realized, but more tools are needed. He finds that integration is difficult. The standard of openEHR is valuable and should be adopted more broadly and could be applied for the new Dutch standard for ZIBs, care information blocks.

3. CASE CO3

In this healthcare organization all CI principles have been observed and are applied. The functional manager offers details of their method of working with the different conceptual models, called archetypes in openEHR. He states that functionality is easy to adapt, when looking at the flexibility of the conceptual models:

ForCI-1:

> "We have a library of archetypes, based on the standard. We do not use a lot of them. Some are for hospitals. They register different attributes than we do. We can download the conceptual models and use them in the application. Yes, we can download data schemas and then add functionality to them."

Functionality can be added in the following way:

> "The data schemas can be extracted from the (available) tool, there are many archetypes in stock. We can select one and make forms for them. We can also edit the data schemas with a dedicated tool for archetypes."

The principle CI-2 is seen in the forms, where the organization can develop forms itself. Forms determine what users can see on the screen, the user interface.

The power of this way of working lies in flexibility of programming code and the collaboration with the supplier of the openEHR software.

> *"We receive the benefits of the standard by working together with the supplier of the openEHR software in developing new functionality, but we have not decided beforehand that we wanted software based on this standard. The supplier of the openEHR software knows what the benefits are for development and can apply the standard to make functionality rapidly and relatively less expensive."*

4. CASE CO4

The reason why openEHR software has been selected in this organization is to enable external access to their systems and collaboration with other institutes.

The functional manager states:

> "The characteristic of openEHR, that we can run and process different models, is not recognizable in other software. Only in the openEHR software."

And he follows up with:

> "We are capable of making our own models, and the software supports different models. Low level reuse, components of the supplier of the openEHR software can be used in other systems, such as EHR, but not the other way around."

The organization is not satisfied with the flexibility of the IT infrastructure in its current state, e.g., when looking at the modularity of the IT infrastructure and ease of replacing applications by other applications:

> "No modularity, in the sense that components can be isolated and replaced. Changes do affect the IT infrastructure. The modularity of our systems is not very high. I would score a 3 on a scale of 7."

He adds that problems arise when you must migrate data, despite using standards.

5. CASE CO5

In case 5 all the CI principles have been recognized and five of these are applied. The IT architect mainly discusses the flexibility of different IT applications as seen from a technical perspective.

The IT architect notices the potential of openEHR software for increasing flexibility and integration in the IT infrastructure. Functionality needed by the organization is offered by the openEHR software but not by the EHR system that the organization deployed before.

He observes IT flexibility in other software too:

> "Other new software has possibilities that are lacking in old software, e.g. the internet, the openEHR software modules and Topdesk. New software is better structured and built."

In contrary to case CO4 he observes that openEHR also offers the possibilities for integration, it aids in extracting data for reuse elsewhere. Data from the (old) EHR-system can be presented in the openEHR system seamlessly. CI principle 2 is mentioned specifically, because it is a substantial part of openEHR software.

Then he adds that extra flexibility to adapt to different target groups of users is typical for openEHR:

> "In a standard EHR there exists one process for all care workers and it is individualistic. We more often work in groups. In the openEHR software you can serve different groups with the same information."

9.6. Summary

In this section we find that openEHR organizations differ in working with CI principles. The similarities and differences will be discussed in the concluding section.

9.6.1. Research Question 2: IT Outsourcing with CI

In the next stage the interviews have been analyzed for statements about IT outsourcing. The results that are summarized in the table are interpretations based on statements of respondents. We searched in the qualitative data for all statements that mentioned how organizations view the goals of IT outsourcing and the strategy for achieving the goals.

Table 9.6. Attributes of IT outsourcing configurations

Capability Factors	**Attributes of ITO Configurations**	**CO1**	**CO2**	**CO3**	**CO4**	**CO5**	**CN6**	**CN7**	**CN8**	**CN9**	**Total(5) open-EHR**	**Total(4) Non-open-EHR**
Service Level Strategy	Outsourced: Development of applications	√	√	√	√	√	√	√	√	√	5	4
	Outsourced: Development of integration between applications	√	√	√	√	√	√	√	√	√	5	4
	Closed data models						√	√	√	√	0	4
	Outsourced: IT architecture function	N	N	N	N	N	N	N	N	N	0	0
	Outsourced: IT management function	N	N	N	N	N	N	N	N	N	0	0
	Outsourced: IT functional management	N	N	N	N	N	N	N	N	N	0	0
	Collaboration with suppliers about functionality of applications	√	√	√	√	√					5	0
Supplier Strategy	Best of Breed	√		√							2	0
	Best of Suites					√	√				1	1
	Based on Enterprise Architecture	√	√	√		√	√	√	√		3	4
Governance Factors												
Commercial Relationship	Direct contact with supplier of openEHR	√	√	√	√	√					5	0
	User groups mainly						√	√	√	√	0	4
	User groups and extra contracts						√		√	√	0	3

√: statements support the presence of the attribute, N: statements deny the presence of the attribute Empty cell: no explicit statements about the attribute have been expressed.

9.7. Service level strategy

In the table the first topic concerns the service level, how large is the proportion of IT functions and IT resources that have been outsourced. All organizations have outsourced the software development and maintenance function. They purchase COTS software to support the care organization. IT strategy, IT architecture and IT management are not outsourced and are executed by the IT architects and IT departments in the organization.

Differences exist between de openEHR organizations and the non-openEHR organizations regarding EHR systems with an open or closed data model and in the collaboration with the suppliers.

In statements of openEHR organizations we find that there is openness of data models in the software, although the database tables are not always understandable, due to the meta-model that is implemented. For the openEHR organizations the statement of CO1 illustrates application of data models:

> "The manner in which the components of our information systems are organized and integrated allows for rapid changes, we can change the system fast with openEHR, we can change forms to work with data ourselves. Difficulties with openEHR are encountered when we reach the information layer and the (Archetype) standard, extra effort is necessary. But it can be done."

The non-openEHR organizations view the data model as a responsibility of the supplier. A remark in case CN6:

> "If you want to adapt functions and data structures in the EHR, you have a problem, because that is what is very difficult for the suppliers."

Case CN9 confirms the former statement:

> "We have no knowledge of the extent and techniques for data consistency."

9.8. Supplier strategy

The organizations differ in strategies to select suppliers in achieving a specific IT architecture. Often mentioned are best of breed or best of suites. Almost all organizations apply an enterprise architecture with a detailed IT architecture to aid the IT architects in their purchasing decisions. We did not find consistent differences between groups of organizations.

9.9. Governance factors

In governance factors we see similarities between all organizations. They all mention the suppliers' efforts to implement adaptations in software required by governmental regulations and laws. They evaluate implementations as satisfactory. However, there are also notable differences between de openEHR organizations and the non-openEHR organizations with regard to the relationship with the suppliers. Examples of differences can be found in procedures, consultations and (costs for) requests for change.

All openEHR cases describe a close collaboration with the supplier of the openEHR software. The non- openEHR organizations comment on the slow procedures that have been experienced.

Examples of statements of openEHR organizations are, for CO1:

> ”When we have questions or change requests for conceptual models, new archetypes must be developed together with the supplier of the openEHR software. This is possible, we do not have the need often, but it has happened.”

The current situation differs from the one with the former supplier, according to CO1:

> “In the old EHR system every interconnection has to be developed separately and paid for separately.”

Despite realized customization the standard software could be even more flexible, according to CO2:

> “Customization in software is added (by the openEHR supplier), but maybe we could have achieved the same result with better modeling. We are examining this with the supplier of the openEHR software. We see great potential, perhaps we could solve 80/90 percent of our requirements with standardization.”

In the third case, CO3, communication with the supplier made them realize the value of the standardof openEHR, see Case description CO3.

Fast communication is appreciated by CO4:

> “We notice that if we have a question or problem, we immediately get a reaction from another mental health care organization with openEHR or the supplier of the openEHR software.”

The last IT architect of the openEHR organizations describes how the organization will lessen the dependence on the supplier, CO5:

> “At this moment the gap is too big, so we extract all the information from the current database of the EHR and store it in the openEHR data base. Later we will take steps to migrate the financial registration from the old EHR to openEHR. We already apply the invoices functionality from another application. We already migrate necessary data to our data warehouse and then to the external platform. We are becoming less dependent on the supplier of the EHR and will eventually make the step described before.”

For the non-openEHR organizations, the User groups, groups of organizations applying software of the same IS supplier, are a means by which change can be realized. If in these organizations there are requests for change consensus in User groups about adaptations in the software has to be achieved first, before the supplier makes a move. The policy of the supplier requires that changes in the IS have to be used by all affiliated user organizations. Customization of software will lead to bottlenecks in the future. The statements of the non-openEHR organizations describe the role of the User group in case CN7:

“The negotiations of the User group with the supplier of the EHR are important for us. We have to reach consensus to be able to really influence the supplier. Our organization is not involved in a movement for standards for exchanging data.”

Collaboration in the mental healthcare sector is characterized also by CN7:

“Mental health sector is a cohesive sector, with communication and collaboration between different organizations.”

Case CN9 adds:

“We can switch modules on and off in our EHR. But we are working on another design for the EHR with the User group of about 30 other Mental Healthcare organizations.”

Case CN8 adds:

“User groups have their disadvantages, because negotiation takes a long time.”

The relation of the EHR system used and the discussions in User groups is remarked by case CN9:

” This CI principle, flexible conceptual models, probably has not been implemented in our EHR. When I look at the application of our EHR, I can conclude that the system is not very flexible, that it is difficult to change. I base this conclusion on discussions in the User group of mental healthcare organizations and on discussions with the account manager of the supplier. We are part of a niche market and we cannot demand much. there are not many alternatives for our EHR.”

9.10. Discussion and conclusion

9.10.1. RQ 1A: CI principles and openEHR

From Table 9.5. and 6 it becomes clear that the openEHR organizations recognize CI principles in software more than the non-openEHR organizations. They explain how CI principles have been implemented and applied in openEHR software. The non-openEHR organizations encounter the CI principles incidentally. They do discuss the CI principles in the context of flexibility. In other software, for instance in data warehouses, CI principles aid in realizing flexibility in conceptual modeling. The conceptual models can be changed by analysts when needed for categorizing relevant data.

In the table of the openEHR organizations CO1 is less outspoken about the CI principles. The transcript reveals that the organization has implemented several openEHR modules but still uses another Electronic Health Record system for its patient dossier. The latter software package dominates a large part of the IT infrastructure and the respondents are distracted by the characteristics of the current EHR. Moreover, they express minimal involvement with the technical details of the implementation of the IS.

Since the IT infrastructures of all organizations are complex and consist of many applications of different IT suppliers, it could have been more informative if we had drawn up a complete inventory of all systems and its characteristics. However, due to practical and organization specific reasons documenting a complete inventory was not feasible in the current research.

We conclude that, overall, all CI principles have been implemented in openEHR software even though not all respondents have detected the CI principles.

9.10.2. RQ 1B: CI principles and IT flexibility

We have only studied the relation of CI principles and IT flexibility in openEHR organizations. They observe that CI contributes to IT flexibility, but they do not describe a standard approach for working with the openEHR software modules. We compare the statements of respondents with each dimensions of IT flexibility.

For the dimension modularity, three cases, CO1, CO3 and CO5 use the openEHR medical database to integrate the IT applications in a loosely coupled manner. The medical database will decouple the applications by acting as a hub and avoiding point-to-point-connections between single applications. Other respondents of cases CO2 and CO4 speak of tight integration of applications and they encounter difficulties when replacing or outphasing certain applications.

The flexibility of openEHR for IS has been recognized and applied by CO1, CO2, CO3 and CO5.

For the dimension of transparency an effect of CI has been noticed and discussed by many organizations in the interviews. They mention the ease of integration of openEHR software and other software and this is explained by CO1, CO2, CO4 and CO5. Two cases, CO2 and CO4, comment on difficulties in integrating different IT applications, specifically in the process of migration of data. However, case CO2 evaluates the openEHR possibilities as an improvement that did not evolve as far as he would have liked. Standardization proved a topic for the openEHR organizations. One organization, CO3, informed the interviewer that it has not been the first reason for selecting the software, but later they appreciated the flexibility of the IS and contributed the extra flexibility to the openEHR standard. The respondent of case CO2 appreciates working with the openEHR standard because it is dedicated to conceptual models for healthcare. He wants to apply the meta-standard for designing the Dutch Care information blocks standard.

Issues of scalability did not prove to be a relevant issue for the respondents.

The respondents did not mention technical obstacles caused by CI principles in the organization.

We conclude that in the interviews all dimensions except scalability have been mentioned and discussed in relation to CI principles. The openEHR organizations evaluate the IT infrastructure as more flexible than the non-openEHR organizations and they accredit the improvement to CI principles and the collaboration with the supplier of openEHR.

When analyzing the quantitative data we conclude that a difference exists in perceptions of IT flexibility between openEHR organizations and not-openEHR organizations.

The combination of quantitative and qualitative data provides support for the influence of CI principles on transparency. Seamlessly using functionality across the organization and accessibility of the applications to internal and external users have been improved with CI. Reconfiguration of systems is substantially supported, all openEHR organizations confirm that the CI software helps them to adapt IS rapidly. A number of cases, but not all, also mention that CI supports a more modular IT infrastructure and aids standardization.

9.10.3. RQ 2: ITO and IT flexibility

For IT outsourcing common characteristics exist in all nine organizations. They have outsourced the development and maintenance of software. It follows from this strategy that the respondents see themselves as mediators between the care organization and the IT department.

However, there are also remarkable differences in the relations of the organizations to their suppliers. Important differences exist between openEHR and non-openEHR organizations in communication with the IS supplier. The frequency of communication with the openEHR supplier is different in the openEHR organization compared to communication with the EHR systems supplier in the non-openEHR organizations. Moreover, procedure of requests for changes are different

We can summarize the results by stating that the supplier of openEHR has a more intensive contact with the organizations about functionality and adaptations of the conceptual models and application functionality in accordance with the needs of the individual organizations than other suppliers. All openEHR organizations refer to the CI principles in openEHR as the main cause for this flexibility.

The non-openEHR organizations describe the procedures in user groups, where groups of mental health organizations first have to achieve consensus about adaptations in the software before the supplier acts. Customization of software is possible, but the costs for these changes are passed on to the individual organizations. Non-openEHR organizations complain about the closed data models maintained by the IS supplier. The IS suppliers do not provide access to the data models and the possibilities for the organizations to adapt the software, for instance by configurations, are limited.

Summarizing, ITO with CI strengthens the IT flexibility.

In conclusion, when comparing our results to the research of Lee and Kim (Lee & Kim, 1999) we see that the communication with the supplier of the openEHR software becomes part of the IT function. The IT management in the organization consults the supplier often and the communication leads to a more flexible IT in the context of organizations that apply CI. These results confirm the findings that partnership quality affects IT flexibility in a positive way in the context of CI.

9.11. Implications

Theoretically two extremes for implementation of conceptual models in IS can be described. At one extreme the IS is designed for any conceptual model in any domain (CI).

At the other extreme in inflexible software the underlying structure is based on closed, flat database models dedicated to specific business contexts. However, the openEHR standard is positioned in between these extremes. Results show that the principles of CI in openEHR have effects on IT flexibility, but the models require more effort to understand, because of abstractions in concepts. Houy et al. explain how understandability of conceptual models is an ambiguous characteristic in research (Houy, Fettke, & Loos, 2012). Future large- scale research on the comprehension of meta-models will be needed when large scale application of CI principles is desired.

Next, this study finds indications that the concept of modularity in IT infrastructure needs clarification. Questions about modularity lead to contradictory answers in the different organizations. Yoo (Yoo, 2012; Yoo, Henfridsson, & Lyytinen, 2010) thinks that the notion of modularity for flexibility, in general, needs rethinking. He extends the concept of modularity in digital systems with the concept of the layered modular architecture where a digital layer references a physical layer. A layered modular architecture can be found in physical objects with an embedded digital component. Architectures with layers can be extended with subparts and not all subparts need specification beforehand. We have demonstrated the effect of CI on IT infrastructure. With CI, flexible representations of social and physical conceptual models can be realized in IS. Future research could investigate if CI principles can be applied in layered modular architectures.

Finally, Lee and Kim (Lee & Kim, 1999) have found indications that partnership quality has a positive effect on outsourcing success. This study shows that impact of partnerships in contexts where CI has been implemented needs further research, especially regarding contract governance.

9.12. Limitations

When studying the effects of an underlying software structure on IT flexibility more details about the technical characteristics of software could present a more complete image of the direct and indirect effects of software structure on IT processes in the organization. The study of technical implementations, however, encounters difficulties because proprietary software and COTS software limit the possibilities of researchers to study details of software architecture.

For ITO the study has not included an analysis of contracts or contractual agreements between organizations and suppliers. Also, contracts were not discussed in the interviews. Findings will be more complete when contracts can be studied in this context.

Lastly, there are only few suppliers of the openEHR software in the Netherlands. All mental healthcare organizations in this study have consulted the same supplier. Our results seem to be in line with large- scale and longitudinal research on openEHR in Norway. In these studies the collaboration of medical professionals and technical professionals appears crucial for the success of the implementation (Christensen & Ellingsen, 2016; Ulriksen, et al., 2017). In fact Christensen and Ellingsen (Christensen

& Ellingsen, 2016) doubt that a complete separation of the clinical and technological domain is feasible.

They observe that expertise of medical professionals and software technology stays necessary for development of the openEHR models.

Acknowledgment

This research was supported and funded by the HAN University of Applied Sciences.

Appendix

1 Content analysis of semi-structured interviews

Table 9.7. Topics for coding the statements in the interviews

CODE	Name	Description
CI	Conceptual independence	Direct reference to CI and implementation
D	Development	Outsourcing of Software development
ITF	IT Flexibility	Explanation of scores on the items in the questionnaire about IT flexibility
M	Management	Goals of IT Management
PDC	Process oriented DC	Processes and procedures in the organization to manage change in IT
SO	Strategy organization	Orientation of (Care and IT) strategy
V	Supplier	Statement about the supplier
F	Evaluation functionality	Functional requirements alignment with systems
ITO	Outsourcing strategy	Outsourcing strategy

2 Mean scores on IT flexibility questionnaire

Table 9.8. Mean Scores on IT Infrastructure Flexibility Questionnair

Groups of organizations	**Non-openEHR organizations**	**openEHR organizations**
MODULARITY		
1. Our information systems are highly modular	3,0	5,3
2. The manner in which the components of our information systems are organized and integrated allows for rapid changes	1,9	5,0
3. Functionality can be quickly added to critical applications based on end-user requests	2,3	4,0
4. Exchanging or modifying single components does **NOT** affect our IT infrastructure	3,3	3,4
5. Organizational IT infrastructure and applications are developed on the basis of minimal unnecessary interdependencies	4,7	4,8
6. Organizational IT infrastructure and applications are loosely coupled	5,4	3,2
TRANSPARENCY		
7. Remote users can seamlessly access centralized data and processes	6,0	6,4
8. Our user interfaces provide transparent access to all platforms and applications	3,5	6,2
9. Software applications can be easily transported and used across multiple platforms	2,5	5,6
10. Data of one system can be easily used in other systems	2,0	4,9
11. Our firm offers multiple interfaces or entry points (e.g., web access) to external users.	2,8	5,6
STANDARDIZATION		
12. We have established corporate rules and standards for hardware and operating systems to ensure platform compatibility	5,8	6,3
13. We have identified and standardized data to be shared across systems and business units	3,0	5,4
14. Our systems are developed in order to incorporate electronic links to external parties	2,6	5,1
15. Organizational IT infrastructure and applications are highly interoperable	3,0	4,4
16. Organizational IT applications are developed based on compliance guidelines.	4,6	5,4
SCALABILITY		
17. Our IT infrastructure easily compensates peaks in transaction volumes	5,4	5,6
18. Our information systems are scalable	4,3	5,8
19. Our IT infrastructure offers sufficient capacity in order to fulfill additional orders for treatment or diagnosis	5,5	6,2
20. The performance of our IT infrastructure completely fulfills our business needs regardless of usage magnitude	2,6	6,0
TOTAL MEAN	**3,7**	**5,2**

Scores on scale 1 -7 (1 totally disagree, 7 totally agree)

Table 9.9. Mean Scores for Organizations on IT Infrastructure Flexibility

ID – random	IT Flexibility score	Outsourcing Organization-type
1	3,52	Non-openEHR
2	4,44	Non-openEHR
3	3,40	Non-openEHR
4	3,56	Non-openEHR
5	6,70	openEHR
6	4,45	openEHR
7	5,40	openEHR
8	4,50	openEHR
9	5,03	openEHR

Table 9.10. Mean Scores for Groups of Organizations on Dimensions of IT Infrastructure Flexibility

Dimension	openEHR	Non-openEHR
MODULARITY	4,3	3,4
TRANSPARENCY	5,7	3,3
STANDARDIZATION	5,2	3,8
SCALABILITY	5,9	4,5

3 Nonparametric Test

Tests have been executed with SPSS software package with a configuration of significance = 0.05 and confidence interval = .95. The asymptotic significances are displayed.

Table 9.11. Hypothesis Test Summary – Means of IT Flexibility

	Null hypothesis	Test	Sig.	Decision
1	The distribution of IT Flexibility Scores is the same across categories of Groups of Organizations.	Independent-Samples Mann-Whitney U Test	.016[1]	Reject the null hypothesis.

1 Exact significance is displayed for this Table .
**) Groups of organizations are openEHR organizations and non-openEHR organizations

Table 9.12. Hypothesis Test Summary – Means on Dimensions of IT Flexibility

	Null hypothesis	Test	Sig.	Decision
1	The distribution of MOD* is the same across categories of Groups of Organizations**	Independent-Samples Mann- Whitney U Test	.286[1]	Retain the null hypothesis.
2	The distribution of TRANS* is the same across categories of Groups of Organizations**	Independent-Samples Mann- Whitney U Test	.016[1]	Reject the null hypothesis.
3	The distribution of STAND* is the same across categories of Groups of Organizations**	Independent-Samples Mann- Whitney U Test	.063[1]	Retain the null hypothesis.
4	The distribution of SCAL* is the same across categories of Groups of Organizations**	Independent-Samples Mann- Whitney U Test	.063[1]	Retain the null hypothesis.

*) Dimensions are: MOD = Modularity, TRANS = Transparency, STAND = Standardization, SCAL = Scalability

**) Groups of organizations are openEHR organizations and non-openEHR organizations

1 Exact significance is displayed for this Table.

10 Conclusion

10.1. Introduction

This chapter summarizes the conclusions for the main research question. The main research question explores the contribution of IT resources and other IT capabilities to IT flexibility in healthcare organizations. Since IT flexibility is one of the key IT capabilities, we research IT resources and the interdependency between IT capabilities. The main research question is formulated as: *"Which IT resources and other IT capabilities contribute to IT flexibility in healthcare organizations in a changing environment in the Netherlands?"*

The context of a changing environment is crucial in this research because the research questions centered around transformation of existing IT infrastructures for changing demands. First, results for each sub-question are described separately, and then an integrated conclusion of the research is presented. Next, we discuss the contribution of the research study and the limitations of the approach. We conclude with ideas for future research.

10.2. Main findings for research questions

In this section different themes have been distinguished based on the four research questions. Per theme the conclusions of relevant chapters have been summarized. All chapters have contributed to one or two themes.

Theme 1: Models to support enterprise architects.
In this section, we answer our findings for question RQ1: *"Which models can support IT architects in practice with their work of developing new architectures for existing IT infrastructures?"*

The models for supporting enterprise architects that have been designed in collaboration with enterprise and IT architects in practice are described in chapters 2 and 3. These are high-level models.

The high-level model in Ampersand, a rule-based modeling language, and its tool, an engine developed in high-level IT consultancy, seemed promising for fast prototyping of architectures. However, one general model was not sufficient to address different information needs for different goals, and this resulted in different types of models.

The details are described in chapter 3. There were different types of high-level models necessary, depending on the objectives and views of the IT architects.

All high-level models lacked the details of lower-level functional requirements needed by enterprise architects. Thus, the needs of architects coincided with characteristics of axiomatic design (AD) (Suh, 1998), where functional requirements must be mapped to separate IT components on every level of the system. We, therefore, investigated the principle of independence in AD: a mapping of functional requirements to independent software components on all levels.

A new model based on the V-model and AD has been designed. The model has been tested in a thought-experiment with two enterprise architects in one case study. However, although the model seemed capable of bridging the two types, it was not further explored, because the input of data proved very time and labor-intensive.

Conclusion.
The design research in the case studies resulted in enterprise architecture (EA) models in which high-level requirements are mapped to high-level IT applications and IT components in IT infrastructure. The high-level models supported the enterprise architects with managing complexity and consistency. However, models lacked a mapping for lower-level functional requirements to lower-level IT components that were necessary for decision making about which parts of IT components to change and which to keep unaltered. A model based on axiomatic design (AD) and the V-model proved capable of mapping requirements on any level. However, the input for the model was too time and effort-intensive and thus not suitable for the enterprise architects. Thus, we conclude that it was not feasible to implement the new model in the organizations.

Theme 2: State of affairs regarding IT flexibility in healthcare.
In this section we answer our findings for question RQ2: *"What is the state of affairs concerning the flexibility of IT architectures in large IT infrastructures in healthcare organizations in the Netherlands?"*

For this research question we refer to chapter 4 and the first part of chapter 6. A further investigation and reflection was conducted on the high-level EA model in chapter 2. An in-depth study of the project reports, meeting reports, and discussions with the leading enterprise architect led to the conclusion that the information systems (IS) and IT infrastructure in this healthcare organization were inflexible. In this organization the EA model could not be applied and implemented because of obstructions and bottlenecks that appeared to come from software structure. Since the software packages and EHR software are commonly used in hospitals, the results suggest that these problems apply to other organizations too.

Then a search for demarcation lines based on AD was performed with IT architects of a medium and large hospital and a home care organization to assess the IT flexibility of large IT infrastructure. This search for demarcation lines related to functional requirements gave unsatisfactory results because the interconnections and dependencies between different IT components impeded the independence of IT components. The findings suggest that the interdependencies are caused by shared use of data on the one hand, such as patient data, data necessary for care processes and workflow data, and technical implementations on the other hand.

Conclusion.
After studying architects with their work of modeling EA, we conclude that formulating consistent high-level models for EA proves feasible, but the implementation of new functionality or changed functionality in existing systems leads to problems. The conclusion is drawn that inflexibility in IT obstructs the work of enterprise architects.

IT inflexibility is related to tight interdependencies between business functionality and technical implementations. The findings suggest that the interdependencies are caused by interdependencies of conceptual models and technical logic in applications.

The IT infrastructures in the case studies have no specific characteristics that make them different from other IT infrastructures in healthcare organizations. Moreover, the applied software in these cases meets with wide-spread acceptance. Therefore, we tentatively state that this problem of high-level independence of IT components and low-level interdependencies can be generalized to other healthcare organizations.

Theme 3: Design principles in software that enable flexibility in IS.
Desk research for RQ3 focuses on software designs that enable the flexibility of IS software. We formulate RQ3: *"Which design principles in software engineering research literature enable flexibility in Information Systems?"*

Because previous chapters show that inflexibility in software systems was a significant obstruction for enterprise architects and IT architects, desk research was conducted to determine factors for improving IT flexibility. The research focused on design principles in theory that are related to functional requirements. The principles of conceptual independence (CI) and axiomatic design (AD) are explained in chapters 5 and 6.

A long line of research literature on the reuse of software confirmed that low-level coding was often needed for implementing new functionality for new requirements. Reasons for inflexibility in IS appeared to be caused by interdependencies in low-level software structures. The principles that influence IS flexibility on a low level of software structure were found in theories of CI (McGinnes & Kapros, 2015) and AD (Suh, 1998). Finally, a design pattern and a model in which the principles are applied are presented in chapter 6. It is presumed that CI can aid in solving the difficulties that emerge in designing software systems based on AD. Since the study on demarcation lines indicates that domain terminology and domain models can be an important cause for the interdependencies between IT components, the presumption is made that decoupling the domain models from the application logic as in CI could improve the flexibility in IS. The final model, as presented in Fig. 10.1. is based on the theories of CI and AD. The model demonstrates that these theories can be combined in software design.

Conclusion.
Theoretical thought-experiments resulted in a model that demonstrates that a combination of principles of AD and CI is possible in IS, see Fig. 10.1. A collection of runtime components each performs a specific business process function as is shown in the model in Fig. 10.1. The components use the terminology in a shared domain model that is specified separately from the application logic, as described

in CI. Each component can run independently from the other components, thereby conforming to the independence principle of AD and the modular systems theory, as described by Simon (Simon, 1962), Schilling (Schilling, 2000), Baldwin and Clark (Baldwin & Clark, 2006). AD adds independence based on functionality to modular systems theory. Thus, following AD, each component can also be configured separately and thereby be adapted based on functional requirements.

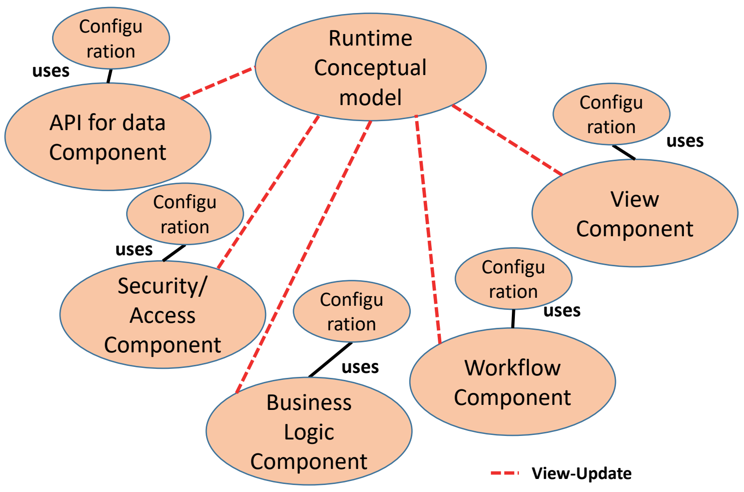


Fig. 10.1. Adaptable information system with system and business functions.

The principles we have found in the existing IS research literature that do realize IS flexibility based on functional requirements are the design principles of AD and CI.

Theme 4: Contribution of conceptual independence and outsourcing to IT flexibility.

The last question, RQ4, returns to the study of IT flexibility in practice: *"Do the implementation of Conceptual Independence and IT outsourcing affect the IT infrastructure flexibility of the organization in healthcare?"*

The theory claims that CI appeared to be an important factor in IS flexibility and a further study has been executed. It is presumed that CI has not been implemented in state-of-the-art software for healthcare systems.

Results have been described in chapters 7, 8 and 9. First, the presumption that CI has not been implemented is studied in open-source software for healthcare. The results are presented in chapter 7. Two out of three systems adventitiously and scarcely apply the design principle CI, in which a separation is made between the conceptual models and terminology and the application logic. Remarkably one system does show the characteristics of the principle of CI. This application implements the openEHR specification. Since the implementation of openEHR is rare in healthcare systems in the Netherlands, we can conclude that CI is not commonly applied in EHR software. After the open-source study an empirical study has been performed to test the effects of CI on the flexibility of organization-wide IT infrastructure in organizations that have applied openEHR. Results are described in chapter 8.

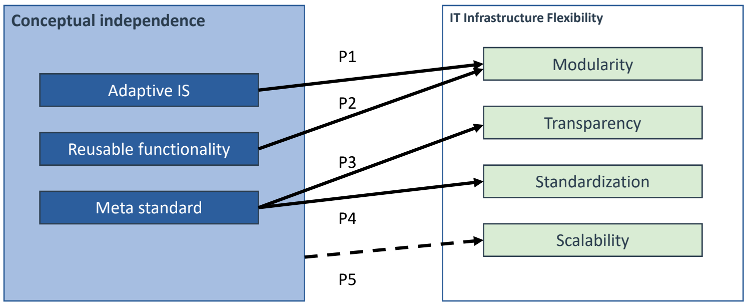


Fig. 10.2. Framework of the effects of CI on IT infrastructure

In Fig.10.2 the research framework of the influence of Conceptual independence on IT infrastructure flexibility is represented. With the arrows P1 to P4 a positive influence of the implementation of CI on dimensions of IT infrastructure flexibility has been proposed, P5 represents an absence of effects of CI on the dimension of scalability.

The organizations that have implemented CI perceive a positive influence of CI on IT flexibility. The quantitative and qualitative data were consistent.

In a follow-up study the qualitative data were analyzed again in detail to find explanations for how the sub-principles of CI in openEHR software are improving flexibility in software. Statements of IT architects, as analyzed in chapter 9, suggest that IT outsourcing, in combination with CI, greatly influences the IT flexibility of IT infrastructures consisting of data-intensive IT applications.

Conclusion.
We conclude from a study of current open-source EHR software that implementation of the design principle of CI is not common practice in software engineering. When organizations have implemented the design principle of CI in the IT infrastructure, the research demonstrates that the flexibility of the organization-wide infrastructure has improved. Furthermore, in organizations that had implemented CI, the relationship with the IT supplier of openEHR played a substantial role in increasing IT flexibility.

10.3. Discussion and contribution

The different chapters in the thesis each look at the IT infrastructure from a dif-ferent angle. The work of different IT professionals in the organization suggests that the gaps between the different types of IT professionals still exist in practice. The idea that changing IT in the organization can be carried out by IT profession-als who detail models at different levels of abstraction, regardless of technical details, turned out to be untenable. Software is not designed for changing busi-ness requirements. In information systems the bottlenecks come to the surface, other researchers confirm. Based on the results we assume that the flexibility of IT components becomes apparent in IS because in these systems business terms and technical implementation meet.

A recent large-scale study of Ahlemann et al. (Ahlemann, Legner, & Lux, 2020) on value generation through enterprise architecture management (EAM) examines different effects of EAM resources and capabilities on organizational performance. They hypothesize that EA resources and EA modeling, EA planning, EA implementation and EA governance will influence the organizational performance as a second-order factor. Value generation will primarily be increased by IS resources and IS capabilities. The study, published in 2020, reports findings of 8 case studies of large companies in different industry sectors, varying from retail and banking to food and government. The companies managed large scale IT infrastructures for which EA could be suitable. Ahlemann et al. confirm the hypothesis that EAM as a second-order effect on value generation works primarily through IS resources and IS capabilities. A similar result has been reported by Pattij et al. (Pattij, van de Wetering, & Kusters, 2019). Other studies focus on dynamic enterprise capabilities to explore theoretical foundations for EA (Van de Wetering, 2020). We select the work of Ahlemann for positioning our study because it covers the broad field of enterprise architecture and IS and shows recent empirical data of large organizations that have confirmed the role of IS capabilities.

In this context our contribution can be positioned as a more detailed investigation of the relation between the work of enterprise architects and the enabling or impeding influence of the IS infrastructure. Ahlemann et al. state that the accepted ideas that EA modeling and documentation are sufficient and helpful in adapting IT and aligning B/IT were not confirmed in the study (Ahlemann, et al., 2020). This finding is consistent with the results in our studies.

Where Ahlemann et al. (Ahlemann, et al., 2020) demonstrate that primarily IS resources and IS capabilities as high-level factors create value for firms, this dissertation shows that underlying software structure plays a role in IS capabilities by increasing IT infrastructure flexibility. The current research on EAM does not take the low-level software development capabilities into account. Moreover, effects of outsourcing the IS development function on IT flexibility are not mentioned by Ahlemann et al. (Ahlemann, et al., 2020).

This dissertation, however, describes mechanisms such as the role of implementations of CI in creating flexibility in software, thereby leading to opportunities to change the business functionality of applications without reprogramming complete IS because domain models can be changed separately. Flexible software forms the foundation for the execution of higher-level processes of EAM that can benefit the organization. If mechanisms for flexibility in software, such as CI, are missing or if inflexibility has unintentionally been built in the software, then an IS resource can profoundly oppose needed change.

This research contributes knowledge about the impact of low-level decisions made in software design on high-level architecture practices. Starting from a high-level view on IT infrastructure, this study drills down to low-level software structures. High-level enterprise architectures in case studies could not be applied in organizations due to a lack of information about the mapping of functional requirements to software components. In the existing research, knowledge about the effects of low-level design decisions on the adaptability of IS for future requirements seemed incomplete. A study about the influence of low-level software development on organizations' IT flexibility has not been executed before.

This study reconfirms the gap between a high-level view of IT infrastructure as seen from the care organization and the technical IT management view as seen from IT departments. This study adds to the knowledge of the effects of underlying software structures that concern technical and functional aspects of IT applications and can contribute to knowledge that bridges the different views on IT.

10.3.1. Overall conclusion

The outcomes of the different case studies lead to the conclusion that the IT resource of IS software brings forth substantial effects on IT flexibility. In the first case studies, the underlying software structure has disclosed itself in obstructions of IT flexibility. Later studies demonstrate that a software structure based on CI has a positive impact on the flexibility of the IT infrastructure beyond the scope of a single application.

We, therefore, see CI as an enabler for IT flexibility. The implementation of CI as a design principle leads in organizations to a rethinking of the structure of the IT infrastructure as a whole. The value of CI as an enabler implies that software design needs to take a different approach toward the implementation of conceptual models in IS. The software itself will be structured in such a way that it can operate on meta-models instead of on hard-coded domain models. Consequences for education arise because students need to understand the value of flexibility in business models for business and the consequences for resulting data models. More research is needed to determine an optimal degree of flexibility in conceptual models. Complete flexibility could be too far-reaching, leading users to program their own applications.

Furthermore, it has been observed that the outsourcing of the software development function has consequences for IT management. When the creation of new functionality and maintenance of software in IS shifts to one or more IS suppliers, the result is a greater dependency on the IS suppliers in case of new demands on the organization. Systems based on CI change the relationship between the outsourcing organization and the IS supplier and can lead to a collaboration that improves IT flexibility.

This study emphasized the IT resource of underlying software structure as an enabling or impeding factor for IT flexibility and suggests that attention for this aspect in organizations will exert an influence on other IT resources and IT capabilities, such as IT outsourcing.

We hypothesize that, when IT flexibility can be enhanced, the work of enterprise architects will be facilitated and their capability to adapt components in an existing IT infrastructure to new demands in the environment will increase.

10.4. Limitations

Due to the novelty of the research domain the research in this dissertation was conducted in an explorative way. Research questions were formulated in an ongoing process. The research has been executed in stages, where each stage was planned after results of the previous study had been analyzed. There were limitations in case studies concerning access to applications in IT infrastructures. Therefore, the scope of the research questions that could be examined was constrained. For instance, direct research of IT infrastructures and commercial IT applications was not possible.

The cases were selected from the network of the researchers or from related networks. Even when access to organizations had been approved of, it was difficult to attain access to information that was related to the current IT operations. IS infrastructures were difficult to examine in detail due to the confidentiality of data.

We have not included the organizational aspects and interests of stakeholders, such as goals related to the costs and benefits of the organization on a meso economic scale, because the early results drew attention to the role of software structure and later research confirmed that questions about software structure remained.

External factors that were encountered in the case studies have not been extensively researched. First, the complexity of the task of the enterprise architects and the mainly top-down approach could have been a source for bottlenecks. The developments in healthcare in combination with the volatile technological developments, could be too much for an enterprise architect to manage.

Secondly, the commercial and financial interests of IT suppliers have not been considered, e.g., the financial stimuli to apply and adapt legacy systems instead of developing state of the art new technology. Attempts to interview firms on these aspects proved problematic. Since sales and marketing of firms depend on the image of expertise, a collaboration with firms for research that potentially leads to a critical view on the IT sector will be difficult to achieve.

We have not made a comparison between different professional sectors. All research has been performed in healthcare organizations because in this sector the necessity of IT infrastructure change had been evident and our network provided access.

10.5. Future research

The study opens up new themes for future research, for instance, the inclusion of software structure in the business perspective gives rise to questions about the degree of modularity necessary for IT flexibility, questions about the structure of the meta-models that are applicable for business data and questions about the organization of the modeling processes. All the IT capabilities related to meta-models and software structure must eventually contribute to the dynamic capabilities of the organization to survive in changing environments.

Modularity. The important dimension of IT flexibility, modularity, needs further clarification. We have found that the decomposition of IT infrastructures in independent components that are responsible for high-level functionality obstructs changes in software on a more detailed level. The relation between functionality and low-level IT components has meaning and value for the organization, a relation that has been ignored by IS suppliers and healthcare organizations in practice.

Meta-model for healthcare standards. The thesis opens interesting research themes concerning the effects that the application of meta-models has on the organization and on technical change in the IT infrastructure. It may even lead to a less top-down approach for enterprise architects. The managing and application of meta-models involves more than the IT capabilities and relates to the dynamic capabilities of organizations. Concrete questions that evaluate current healthcare standards such as FHIR, HL7 and the emerging standard ZIB in the Netherlands should be seen in the light of flexibility of domain models and thus of requirements for meta-models for healthcare standards such as openEHR.

Modeling in the organization. The application of meta-models for software structure requires attention in the organization for developing the conceptual models. Modeling will need to be the responsibility of the organization as a whole and the IT will be requested to support the activities of modeling and the managing and application of models in software. Research suggests that new processes and governance have to be set up for the modeling processes. These modeling processes and their relation to dynamic capabilities are a promising area of research.

Top-down research. In this research, the approach of enterprise architects in the study can be characterized as a top-down approach. The top-down approach in the cases came to a standstill at the moment it encountered low-level implementations in software. In future research the top-down approach itself could be subject of research in its effects on agility and dynamic capabilities of the organization. Different approaches, top-down, bottom-up and middle-out can be compared in the enterprise architecture domain. The existing research on the relations between IT flexibility and agility (Ahsan & Ngo-Ye, 2005) and IT flexibility and dynamic capabilities (Mikalef, Pateli, & Van de Wetering, 2016; Pavlou & El Sawy, 2011) can thus be enriched.

Inclusion of CI in software design. The need for flexible conceptual models has been a primary finding in this research. The role of CI in enabling IT flexibility, as seen from the organizations, implies that software can and should be structured in a different way. Hitherto, designing of software has been the sole responsibility of the IT supplier. Inclusion of CI in software designs changes the practice of development of IS. There are implications for education where the principle of CI can be taught and where experiments and studies can be performed with methods for implementation and development of CI based applications.

10.6. Reflection

At the beginning of the project I started with questions about the development and transformation of complex IT systems and their application in organizations. Conflicting newspaper messages reported about innovations of architecture methods and new development methods for IT architectures and, at the same time, described unforeseen, far-reaching difficulties when replacement of legacy IS had become necessary. Since I had been employed as a developer and consultant for large scale IS in telecom organizations for five years from 1998-2003, I was curious about how enterprise architectures functioned in real-life organizations after 2010 and the degree of business/IT alignment that had been achieved. The research questions that were formulated concerned the enterprise architects and their work. In the first three cases, I experienced the gap between the technical IT professionals and the enterprise architects. In practice, differences in technical views of IT infrastructures and business views of IT could not be ignored.

My experience in teaching Java programming for enterprise systems had prepared me for differences in viewpoints of IT professionals, but the reality showed that these differences were more serious than expected and led to misunderstandings and profoundly wrong images about the work and ideas of the other party. The observations and statements of interviewees could in all cases only be meaningful within the context and IT domain that they referred to.

IT professionals from business and technical departments evaluated the enterprise architecture and IT infrastructure reliably and consistently. We see their interpretations and statements as valid statements about the organizational IT. We as researchers could assess reliability because the IT professionals showed extended knowledge regarding architecture and IT, volunteered extensive examples demonstrating the need to align healthcare and IT, gave extensive descriptions of well and not well-functioning IT applications and spoke of successful and unsuccessful attempts to change the infrastructure and the IS. The statements and discussions were consistent over time and in line with the literature.

My experience with IT issues in the work of students taught me that there needed to be a human side to both views. The IT technical professional approached systems in a human manner, trying to maintain control of systems that were often realized and programmed by others. Developers tried to create software code that was suitable, interesting and beautiful, but could not always succeed in creating it because of economic or other reasons. In the 20 years that I have taught java programming, the enthusiasm of students and their motivation to create new products were striking.

However, the criteria for evaluation of systems differed for technical IT professionals and for users of the systems. Care professionals needed access to data in such a way that the information could be interpreted for the medical or care practice. Even the criteria for quality differed between these groups.

In practice tremendous effort is spent by technical and modeling experts trying to research the thinking models/conceptual models of care professionals to incorporate these in systems that in the next stage were perfected in a technical sense. The effort was aimed at making conceptual models as perfect and stable as possible.

However, during the Ph.D. project, it became more and more clear to me that flexibility in thinking of medical and care professionals had been the blind spot of technical IT professionals. Then the question popped up whether flexibility in conceptual models could be realized in real-life systems at all. During the course of the study I struggled with this issue, even though it was apparent that CI as a design principle has been realized in systems that are ubiquitous. We can think of office applications, such as word processors, databases, spreadsheets and folders and directory systems. All these applications are domain independent. Data models originating from different domains can and are implemented without causing the applications to falter. However, surprisingly for users, conceptual models were not seen as variables in electronic patient dossiers.

The trace that I followed in the project has been interesting and puzzling. The project of the Ph.D. brought many surprises and valuable moments. The most inspiring hours were spent in contact with architects and IT experts or researchers that were willing to discuss their ideas. At this moment it is unpredictable whether the adaptability of software applications will be realized with CI in the future. My personal impression is that many of the enthusiastic and inspired IT professionals that I have encountered support design solutions along these lines.

References

Aanestad, M., Grisot, M., Hanseth, O., & Vassilakopoulou, P. (2017a). Information infrastructures and the challenge of the installed base *Information Infrastructures within European Health Care* (pp. 25-33): Springer, Cham.

Aanestad, M., Grisot, M., Hanseth, O., & Vassilakopoulou, P. (2017b). Introduction: Information infrastructures within European health care: Working with the installed base *Information infrastructures within European health care: Working with the installed base* (pp. 1-10): Springer.

Ahlemann, F., Legner, C., & Lux, J. (2020). A resource-based perspective of value generation through enterprise architecture management. *Information & Management*, 103266.

Ahmad, A., Jamshidi, P., & Pahl, C. (2013). A framework for acquisition and application of software architecture evolution knowledge: 14. *ACM SIGSOFT Software Engineering Notes, 38*(5), 1-7. doi: 10.1145/2507288.2507301

Ahsan, M., & Ngo-Ye, L. (2005). *The Relationship Between IT Infrastructure and Strategic Agility in Organizations.* In AMCIS 2005 Proceedings, (pp. 266).

Aier, S. (2014). The role of organizational culture for grounding, management, guidance and effectiveness of enterprise architecture principles. *Information Systems and E-Business Management, 12*(1), 43-70.

Ajienka, N., Capiluppi, A., & Counsell, S. (2017). *Managing Hidden Dependencies in OO Software: a Study Based on Open Source Projects.* In 2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), (pp. 141-150), Toronto. IEEE.

Alexander, C., Ishikawa, S., & Silverstein, M. (1977). *A pattern language: towns, buildings, construction* (Vol. 2): Oxford University Press.

Aryani, A., Perin, F., Lungu, M., Mahmood, A. N., & Nierstrasz, O. (2011). *Can we predict dependencies using domain information?* Paper presented at the Reverse Engineering (WCRE), 2011 18th Working Conference on.

Atalag, K., Yang, H. Y., & Warren, J. (2014). Assessment of software maintainability of openEHR based health information systems - A case study in endoscopy. *International Journal of Medical Informatics*(83), 849-859.

Bachmann, F., Bass, L., Garlan, D., Ivers, J., Little, R., Merson, P., . . . Stafford, J. (2011). *Documenting Software Architectures: Views and Beyond*: Addison-Wesley Professional.

Baldwin, C. Y., & Clark, K. B. (2006). Modularity in the design of complex engineering systems. *Complex engineered systems*, 175-205.

Barney, J. B. (2001). Resource-based theories of competitive advantage: A ten-year retrospective on the resource-based view. *Journal of management, 27*(6), 643-650.

Bass, L., Clements, P., & Kazman, R. (2012). *Software architecture in practice* (Third Edition ed.). Upper Saddle River, NJ: Addison-Wesley Professional.

Beale, T. (2002). *Archetypes: Constraint-based domain models for future-proof information systems.* In OOPSLA 2002 workshop on behavioural semantics, (pp. 1-69).

Beale, T. (2002, 2003). openEHR / ISO 18308 Conformace Statement Retrieved 25 september 2020, 2020, from https://specifications.openehr.org/releases/0.9/requirements/iso18308_conformance.pdf

Beale, T., & Heard, S. (2007). Archetype Definitions and Principles Retrieved 10 feb 2020, 2020, from https://specifications.openehr.org/releases/1.0.2/architecture/am/archetype_principles.pdf

Bézivin, J., & Gerbé, O. (2001). *Towards a precise definition of the OMG/MDA framework*. Paper presented at the Automated Software Engineering. (ASE 2001). 16th Annual International

Bhérer, L., Vouligny, L., Gaha, M., Redouane, B., & Desrosiers, C. (2015). *Ontology-based adaptive information system framework.* In Proceedings of the SEMAPRO 2015: The Ninth International Conference on Advances in Semantic Processing (Nice, France), (pp. 110-115).

Bocij, P., Greasley, A., & Hickie, S. (2008). *Business information systems: Technology, development and management*: Pearson education.

Boonstra, A., Versluis, A., & Vos, J. F. (2014). Implementing electronic health records in hospitals: a systematic literature review. *BMC health services research, 14*(1), 370.

Broadbent, M., Weill, P., & Neo, B.-S. (1999). Strategic context and patterns of IT infrastructure capability. *The Journal of Strategic Information Systems, 8*(2), 157-187.

Bui, Q., Leo, E., & Adelakun, O. (2019). Exploring complexity and contradiction in information technology outsourcing: A set-theoretical approach. *The Journal of Strategic Information Systems, 28*(3), 101573.

Buschmann, F., Henney, K., & Schmidt, D. (2007a). *Pattern-oriented Software Architecture: on patterns and pattern language* (Vol. 5): John wiley & sons.

Buschmann, F., Henney, K., & Schmidt, D. C. (2007b). *Pattern-Oriented Software Architecture, Volume 4: A Pattern Language for Distributed Computing*. Chichester, UK: Wiley.

Bygstad, B., Hanseth, O., & Truong Le, D. (2015). *From IT silos to integrated solutions: a study in e-health complexity* In ECIS 2015 Completed Research Papers, (pp. Paper 23).

Byrd, T. A., & Turner, D. E. (2000). Measuring the flexibility of information technology infrastructure: Exploratory analysis of a construct. *Journal of Management Information Systems, 17*(1), 167-208.

CGI Groep. (2016). SMART Process Acceleration Development Environment Retrieved 13-1-2017, 2017, from https://en.wikipedia.org/wiki/SMART_Process_Acceleration_Development_Environment

Chan, Y. E., Sabherwal, R., & Thatcher, J. B. (2006). Antecedents and outcomes of strategic IS alignment: an empirical investigation. *Ieee Transactions on Engineering Management, 53*(1), 27-47.

Chanopas, A., Krairit, D., & Ba Khang, D. (2006). Managing information technology infrastructure: a new flexibility framework. *Management Research News, 29*(10), 632-651.

Chiu, C., Kohli, A. S., & Chaczko, Z. (2012). Building an Intelligent Health System Using an Evolutionary Architectural Model of Middleware. *International Journal of Advanced Computer Science, 2*(2), 56-64.

Cios, K. J., & Moore, G. W. (2002). Uniqueness of medical data mining. *Artificial intelligence in medicine, 26*(1-2), 1-24.

Cleve, A., Gobert, M., Meurice, L., Maes, J., & Weber, J. (2015). Understanding database schema evolution: A case study. *Science of Computer Programming, 97*, 113-121. doi: https://doi.org/10.1016/j.scico.2013.11.025

Cordis EU. (2014). The Good European Health Record (GEHR) Retrieved 27 august 2020, 2020, from https://cordis.europa.eu/project/id/A2014

Creswell, J. W., & Creswell, J. D. (2005). Mixed methods research: Developments, debates, and dilemmas *Research in organizations: Foundations and methods of inquiry* (pp. 315-326).

Czarnecki, K. (2005). Overview of generative software development *Unconventional Programming Paradigms* (pp. 326-341): Springer.

De Reuver, M., Sørensen, C., & Basole, R. C. (2018). The digital platform: a research agenda. *Journal of Information Technology, 33*(2), 124-135.

Dennis, A. R., & Garfield, M. J. (2003). The adoption and use of GSS in project teams: Toward more participative processes and outcomes. *Mis Quarterly*, 289-323.

Dibbern, J., Goles, T., Hirschheim, R., & Jayatilaka, B. (2004). Information systems outsourcing: a survey and analysis of the literature. *ACM SIGMIS Database: the DATABASE for Advances in Information Systems, 35*(4), 6-102.

Do, S.-H., & Suh, N. P. (1999). Systematic OO programming with axiomatic design. *Computer, 32*(10), 121-124.

Duncan, N. B. (1995). Capturing flexibility of information technology infrastructure: A study of resource characteristics and their measure. *Journal of Management Information Systems, 12*(2), 37-57.

Eisenhardt, K. M., & Martin, J. A. (2000). Dynamic capabilities: what are they? *Strategic management journal, 21*(10-11), 1105-1121.

Ferreira, H. S., Correia, F. F., Aguiar, A., & Faria, J. P. (2010). Adaptive object-models: A research roadmap. *International Journal On Advances in Software, 3*.

Fetters, M. D., Curry, L. A., & Creswell, J. W. (2013). Achieving integration in mixed methods designs—principles and practices. *Health services research, 48*(6pt2), 2134-2156.

Fowler, M. (2002). *Patterns of enterprise application architecture*: Addison-Wesley Longman Publishing Co., Inc.

Franke, U., Ekstedt, M., Lagerström, R., Saat, J., & Winter, R. (2010). *Trends in enterprise architecture practice–a survey.* In International Workshop on Trends in Enterprise Architecture Research, (pp. 16-29). Springer.

Garde, S., Knaup, P., Hovenga, E. J., & Heard, S. (2007). Towards semantic interoperability for electronic health records. *Methods of Information in Medicine, 46*(03), 332-343.

Garlan, D., Allen, R., & Ockerbloom, J. (1995). Architectural mismatch: Why reuse is so hard. *Ieee Software, 12*(6), 17-26.

Garlan, D., Allen, R., & Ockerbloom, J. (2009). Architectural mismatch: Why reuse is still so hard. *Ieee Software, 26*(4), 66.

Goeminne, M., Decan, A., & Mens, T. (2014). *Co-evolving code-related and database-related changes in a data-intensive software system.* In 2014 Software Evolution Week-IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE), (pp. 353-357), Antwerp. IEEE.

Gondelach, S., Levering, W., Van der Meer, J., Bot, H., Oord, E., Cillessen, F., & Schat, B. (2010). *Beter met architectuur*: Van Haren Publishing.

Gotel, O. C., & Finkelstein, A. C. (1994). *An analysis of the requirements traceability problem.* In Requirements Engineering, 1994., Proceedings of the First International Conference on, (pp. 94-101). IEEE.

Gregor, S. (2006). The nature of theory in information systems. *Mis Quarterly*, 611-642.

Haitzer, T., Navarro, E., & Zdun, U. (2017). Reconciling software architecture and source code in support of software evolution. *Journal of Systems and Software, 123*, 119-144. doi: https://doi.org/10.1016/j.jss.2016.10.012

Hanseth, O., Monteiro, E., & Hatling, M. (1996). Developing information infrastructure: The tension between standardization and flexibility. *Science, Technology, & Human Values, 21*(4), 407-426.

Hardcastle, E. (2011). *Business Information Systems*: Bookboon.

Hellberg, S., & Johansson, P. (2017). eHealth strategies and platforms–The issue of health equity in Sweden. *Health Policy and Technology, 6*(1), 26-32.

Henderson, J. C., & Venkatraman, N. (1993). Strategic alignment: leveraging information technology for transforming organizations. *IBM Syst. J., 32*(1), 4-16. doi: DOI: 10.1147/sj.382.0472

Herraiz, I., Rodriguez, D., Robles, G., & Gonzalez-Barahona, J. M. (2013). The evolution of the laws of software evolution: A discussion based on a systematic literature review. *ACM Computing Surveys (CSUR), 46*(2), 28. doi: 10.1145/2543581.2543595

Hevner, A., & Chatterjee, S. (2010). Design science research in information systems *Design Research in Information Systems: Theory and Practice* (Vol. 22 Integrated Series in Information Systems, pp. 9-22). Boston, MA: Springer.

Hevner, A. R. (2007). A three cycle view of design science research. *Scandinavian journal of information systems, 19*(2), 4.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *Mis Quarterly, 28*(1), 75-105.

Interconfessionele Stichting Gezondheidszorg Rivierenland Culemborg. (2013). Jaardocument 2013 Retrieved 4-4-2017, from https://www.zrt.nl/upload/File/jaarverslagen/JAARDOCUMENT2013.pdf

Jamshidi, P., Ghafari, M., Ahmad, A., & Pahl, C. (2013). *A framework for classifying and comparing architecture-centric software evolution research.* In Software Maintenance and Reengineering (CSMR), 2013 17th European Conference on, (pp. 305-314). IEEE.

Joachim, N., Beimborn, D., & Weitzel, T. (2013). The influence of SOA governance mechanisms on IT flexibility and service reuse. *The Journal of Strategic Information Systems, 22*(1), 86-101.

Joosten, S. (2007). Deriving Functional Specification from Business Requirements with Ampersand. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.118.4137

Kagdi, H., Collard, M. L., & Maletic, J. I. (2007). A survey and taxonomy of approaches for mining software repositories in the context of software evolution. *Journal of Software: Evolution and Process, 19*(2), 77-131. doi: 10.1002/smr.344

Kagdi, H., & Poshyvanyk, D. (2009). *Who can help me with this change request?* In 2009 IEEE 17th International Conference on Program Comprehension, Vancouver. ICPC'09, (pp. 273-277). IEEE.

Kemena, T., Wetering, R. v. d., & Kusters, R. J. (2019). *The impact of IT human capability and IT flexibility on IT-enabled dynamic capabilities*. Paper presented at the Conference: 32nd Bled eConference - Humanizing Technology for a Sustainable Society, Bled, Slovenia.

Kim, G., Shin, B., Kim, K. K., & Lee, H. G. (2011). IT capabilities, process-oriented dynamic capabilities, and firm financial performance. *Journal of the Association for Information Systems, 12*(7), 487.

Kim, Y.-G., & Everest, G. C. (1994). Building an IS architecture: Collective wisdom from the field. *Information & Management, 26*(1), 1-11.

Kotusev, S. (2019). Fake and Real Tools for Enterprise Architecture: The Zachman Framework and Business Capability Model. *Enterprise Architecture Professional Journal*, 1-14.

Labusch, N., Koebele, F., Aier, S., & Winter, R. (2013). *The architects' perspective on enterprise transformation: An explorative study.* In Working Conference on Practice-Driven Research on Enterprise Transformation, (pp. 106-124). Springer.

Lange, M., Mendling, J., & Recker, J. (2012). *A comprehensive EA benefit realization model--An exploratory study.* In System Science (HICSS), 2012 45th Hawaii International Conference on, (pp. 4230-4239). IEEE.

Larman, C. (2005). *Applying UML and patterns : an introduction to object-oriented analysis and design and iterative development*. Upper Saddle River, N.J.: Prentice Hall PTR

Lee, A. S. (1989). A scientific methodology for MIS case studies. *Mis Quarterly, 13*(1), 33-50.

Lehman, M. M. (1996). Laws of software evolution revisited. *European Workshop on Software Process Technology*, 108-124. doi: https://doi.org/10.1007/BFb0017737

Mamlin, B., & Jindal, S. (2017, 18/01/2017). Introduction to OpenMRS Retrieved October 14, 2017, 2017, from https://wiki.openmrs.org/display/docs/Introduction+to+OpenMRS

Mayring, P. (2004). Qualitative content analysis. In U. Flick (Ed.), *A companion to qualitative research* (Vol. 1, pp. 159-176): SAGE Publications Ltd

McGinnes, S. (2011). *The Problem of Conceptual Incompatibility*. Paper presented at the Conference on Availability, Reliability, and Security, Berlin, Heidelberg.

McGinnes, S., & Kapros, E. (2015). Conceptual independence: A design principle for the construction of adaptive information systems. *Information Systems, 47*, 33-50. doi: https://doi.org/10.1016/j.is.2014.06.001

Mendix. (2016). Drive digital transformation and innovation with Mendix aPaas Retrieved 13-1-2017, 2017, from https://docs.mendix.com/

Mens, T., Meurice, L., Goeminne, M., Nagy, C., Decan, A., & Cleve, A. (2017). Analyzing the Evolution of Database Usage in Data-Intensive Software Systems. *2017*(October 14, 2017).

Michels, G., Joosten, S., Woude, J., & Joosten, S. (2011a, 2011/01/01). *Ampersand.* In International Conference on Relational and Algebraic Methods in Computer Science, (pp. 280-293). Springer, Berlin Heidelberg.

Michels, G., Joosten, S., Woude, J. v. d., & Joosten, S. (2011b). *Ampersand applying relation algebra in practice.* In Proceedings of the 12th international conference on Relational and algebraic methods in computer science, (pp. 280-293), Rotterdam, The Netherlands. Springer-Verlag.

Mikalef, P., Pateli, A. G., & van de Wetering, R. (2016). *It Flexibility and Competitive Performance: the Mediating Role of IT-Enabled Dynamic Capabilities*. Paper presented at the European Conference on Information Systems (ECIS2016)

Nachar, N. (2008). The Mann-Whitney U: A test for assessing whether two independent samples come from the same distribution. *Tutorials in quantitative Methods for Psychology, 4*(1), 13-20.

Nictiz. (2017). Zorginformatiebouwstenen Retrieved 18-1-2018, from https://zibs.nl/wiki/HCIM_Mainpage

Object Management Group, I. (2016). Meta Object Facility™ (MOF™) Core 2.5.1 Retrieved October 9, 2017, 2017, from http://www.omg.org/spec/MOF/

Object Management Group, I. (2017). Meta-Modeling and the OMG Meta Object Facility (MOF) . Retrieved October 9, 2017, 2017, from www.omg.org/ocup-2/documents/Meta-ModelingAndtheMOF.pdf

openEHR Foundation. (2020a). openEHR Community Industry Partners Retrieved 24-07-2020, from https://openehr.org/community/industry_partners/

openEHR Foundation. (2020b). What is openEHR? Retrieved 24-07-2020, 2020, from https://openehr.org/about/what_is_openehr

openEMR. (2014). Database structure openEMR Retrieved October 14, 2017, 2017, from http://www.open-emr.org/wiki/index.php/Database_Structure

openEMR. (2017). openEMR-v5.0.0 Retrieved 03/03/2017, 2017, from https://github.com/openmrs/openmrs-standalone

openMRS. (2017a). openMRS Core-v4.0.0 Retrieved 03/04/2017, 2017, from https://github.com/openmrs/openmrs-core

openMRS. (2017b). openMRS Standalone 2.5 Retrieved 03/03/2017, 2017, from https://github.com/openmrs/openmrs-standalone

Őri, D. (2016). *An Artifact-Based Framework for Business-IT Misalignment Symptom Detection.* In IFIP Working Conference on The Practice of Enterprise Modeling, (pp. 148-163). Springer.

Overby, E., Bharadwaj, A., & Sambamurthy, V. (2006). Enterprise agility and the enabling role of information technology. *European Journal of Information Systems, 15*(2), 120-131.

Pattij, M., van de Wetering, R., & Kusters, R. (2019). *From Enterprise Architecture Management to Organizational Agility: The Mediating Role of IT Capabilities.* In Bled eConference, (pp. 31).

Pavlou, P. A., & El Sawy, O. A. (2006). From IT leveraging competence to competitive advantage in turbulent environments: The case of new product development. *Information Systems Research, 17*(3), 198-227.

Pavlou, P. A., & El Sawy, O. A. (2011). Understanding the elusive black box of dynamic capabilities. *Decision Sciences, 42*(1), 239-273.

Pazos, P. (2017). openEHR cabolabs server-v0.9 Retrieved 03/05/2017, 2017, from https://github.com/ppazos/cabolabs-ehrserver

Qiu, D., Li, B., & Su, Z. (2013). *An empirical analysis of the co-evolution of schema and code in database applications*. Paper presented at the Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering.

Ray, G., Barney, J. B., & Muhanna, W. A. (2004). Capabilities, business processes, and competitive advantage: choosing the dependent variable in empirical tests of the resource-based view. *Strategic management journal, 25*(1), 23-37.

Rector, A. L. (1999). Clinical terminology: why is it so hard? *Methods of Information in Medicine, 38*(4/5), 239-252.

Sambamurthy, V., Bharadwaj, A., & Grover, V. (2003). Shaping agility through digital options: Reconceptualizing the role of information technology in contemporary firms. *Mis Quarterly*, 237-263.

Schäfer, W., Kroneman, M., Boerma, W., Van den Berg, M., Westert, G., & Devillé, W. (2010). The Netherlands: Health System Review. In o. b. o. t. © World Health Organization 2010 & E. O. o. H. S. a. Policies (Eds.), *Health Systems in Transition* (Vol. 12): NIVEL.

Schilling, M. A. (2000). Toward a general modular systems theory and its application to interfirm product modularity. *Academy of Management Review, 25*(2), 312-334.

Schmidt, C. (2004). The analysis of semi-structured interviews. In U. Flick (Ed.), *A companion to qualitative research* (pp. 253-258): SAGE Publications Ltd

Simon, D., Fischbach, K., & Schoder, D. (2013). An exploration of enterprise architecture research. *Communications of the Association for Information Systems, CAIS, 32*(1), 1-72.

Simon, H. A. (1962). *The architecture of complexity.* In Proceedings of the American Philosophical Society, (pp. 457-476). American Philosophical Society.

Singh, V., Bhattacherjee, V., & Bhattacharjee, S. (2012). An analysis of dependency of coupling on software defects. *ACM SIGSOFT Software Engineering Notes, 37*(1), 1-6.

Skoulis, I., Vassiliadis, P., & Zarras, A. (2014). *Open-source databases: Within, outside, or beyond lehman's laws of software evolution?* In International Conference on Advanced Information Systems Engineering, (pp. 379-393). Springer.

Star, S. L. (1999). The ethnography of infrastructure. *American Behavioral Scientist, 43*(3), 377-391.

Stichting Carint Reggeland Groep. (2014). Bestuursverslag 2013 Retrieved 4 -04-2017, 2017, from http://www.carintreggeland.nl/dbimages/Bijlagen/Bestuursverslag%202013%20Carintreggeland.pdf

Suh, N. P. (1998). Axiomatic design theory for systems. *Research in engineering design, 10*(4), 189-209.

Suh, N. P. (2001). *Axiomatic Design: Advances and Applications (The Oxford Series on Advanced Manufacturing)*. New York Oxford: Oxford University Press.

Suh, N. P. (2012). Fundamentals of Design and Deployment of Large Complex Systems: OLEV, MH, and Mixalloy. *Journal of Integrated Design & Process Science, 16*(3), 7-28. doi: 10.3233/jid-2012-0001

Suh, N. P., & Do, S.-H. (2000). Axiomatic design of software systems. *CIRP annals, 49*(1), 95-100.

Tafti, A., Mithas, S., & Krishnan, M. S. (2013). The effect of information technology–enabled flexibility on formation and market value of alliances. *Management Science, 59*(1), 207-225.

Tallon, P. P., & Pinsonneault, A. (2011). Competing perspectives on the link between strategic information technology alignment and organizational agility: insights from a mediation model. *Mis Quarterly*, 463-486.

Tamm, T., Seddon, P. B., Shanks, G., & Reynolds, P. (2011). How does enterprise architecture add value to organisations? *Communications of the association for information systems, 28*(1), 10.

Tarenskeen, D. (2016). *Conceptual Independence as an Architecture Pattern for Adaptable Systems.* In ACM Proceedings of the 10th Travelling Conference on Pattern Languages of Programs, (pp. Article 5), Leerdam, AA, Netherlands. Association for Computing Machinery.

Tarenskeen, D., & Bakker, R. (2017). Applying Axiomatic Design and Conceptual Independence in the Domain of IT Systems. *MATEC Web of Conferences, 127(01006).*

Tarenskeen, D., Bakker, R., & Joosten, S. (2013). *Using ampersand in IT architecture*. Paper presented at the 7th International Conference on Research and Practical Issues of Enterprise Information Systems CONFENIS 2013, Prague, Czech Republic.

Tarenskeen, D., Bakker, R., & Joosten, S. (2015). *Applying the V Model and Axiomatic design in the Domain of IT Architecture Practice.* In ICAD 2015, International Conference of Axiomatic Design, Procedia CIRP, (pp. 263-268).

Tarenskeen, D., Van de Wetering, R., & Bakker, R. (2018). Unintended effects of dependencies in source code on the flexibility of IT in organizations. *Communication Papers of the 2018 Federated Conference on Computer Science and Information Systems, 17*, 87-94. doi: 10.15439/2018f93

Teece, D. J., Pisano, G., & Shuen, A. (1997). Dynamic capabilities and strategic management. *Strategic management journal, 18*(7), 509-533.

TheOpenGroup. (2011). *TOGAF Version 9.1 Evaluation copy*: The Open Group.

Thinkwise. (2016). Bedrijfssoftware - Thinkwise Retrieved 13-1-2017, 2017, from https://www.thinkwisesoftware.com/nl/

Tyree, J., & Akerman, A. (2005). Architecture decisions: Demystifying architecture. *Ieee Software, 22*(2), 19-. doi: 10.1109/ms.2005.27

Ulriksen, G.-H., Pedersen, R., & Ellingsen, G. (2017). Infrastructuring in healthcare through the openEHR architecture. *Computer Supported Cooperative Work (CSCW), 26*(1-2), 33-69.

UMC Utrecht. (2012). Directieverslag 2012 Retrieved 12-04- 2017, from http://www.umcutrecht.nl/nl/Over-Ons/Wat-we-doen/Jaarverslag-UMC-Utrecht

Van de Wetering, R. (2019). *Enterprise Architecture Resources, Dynamic Capabilities, and their Pathways to Operational Value*. Paper presented at the International Conference on Information Systems (ICIS) 2019 Munic, Germany. https://aisel.aisnet.org/icis2019/is_development/is_development/6/

Van de Wetering, R. (2020, June 15-17, 2020). *Dynamic enterprise architecture capabilities and organizational benefits: an empirical mediation study.* In ECIS2020, (pp. Marrakech, Morocco. AIS.

Van de Wetering, R., Mikalef, P., & Helms, R. (2017). Driving organizational sustainability-oriented innovation capabilities: a complex adaptive systems perspective. *Current Opinion in Environmental Sustainability, 28*, 71-79. doi: http://dx.doi.org/10.1016/j.cosust.2017.08.006

Van de Wetering, R., Mikalef, P., & Pateli, A. (2017). *How Strategic Alignment Of IT Flexibility, A Firm's Networking Capability, And Absorptive Capacity Influences Firm Innovation*. Paper presented at the The 11th Mediterranean Conference on Information Systems, Genoa, Italy.

Van de Wetering, R., Mikalef, P., & Pateli, A. (2017). *A strategic alignment model for IT flexibility and dynamic capabilities: toward an assessment tool*. Paper presented at the Proceedings of the 25th European Conference on Information Systems (ECIS), Guimarães, Portugal.

Van de Wetering, R., Mikalef, P., & Pateli, A. (2018). Strategic Alignment Between IT Flexibility and Dynamic Capabilities: An Empirical Investigation. *International Journal of IT/Business Alignment and Governance (IJITBAG), 9*(1), 1-20.

Van de Wetering, R., Versendaal, J., & Walraven, P. (2018). *Examining the relationship between a hospital's IT infrastructure capability and digital capabilities: a resource-based perspective*. Paper presented at the Twenty-fourth Americas Conference on Information Systems (AMCIS), New Orleans.

Van den Berg Jeths, A., Timmermans, J. M., Hoeymans, N., & Woittiez, I. B. (2004). Ouderen nu en in de toekomst: 2000-2020. In R. v. V. e. Milieu (Ed.), *Themarapporten van de Volksgezondheid Toekomst Verkenningen en is opgesteld in opdracht van het Ministerie van VWS*.

Van der Raadt, B., Bonnet, M., Schouten, S., & Van Vliet, H. (2010). The relation between EA effectiveness and stakeholder satisfaction. *Journal of Systems and Software, 83*(10), 1954-1969.

Vaquero, L. M., Rodero-Merino, L., & Buyya, R. (2011). Dynamically scaling applications in the cloud. *ACM SIGCOMM Computer Communication Review, 41*(1), 45-52.

Vassiliadis, P., Zarras, A. V., & Skoulis, I. (2015). *How is life for a table in an evolving relational schema? Birth, death and everything in between*. Paper presented at the International Conference on Conceptual Modeling.

Vassiliadis, P., Zarras, A. V., & Skoulis, I. (2017). Gravitating to rigidity: Patterns of schema evolution–and its absence–in the lives of tables. *Information Systems, 63*, 24-46.

Völter, M., Stahl, T., Bettin, J., Haase, A., & Helsen, S. (2013). *Model-driven software development: technology, engineering, management*: John Wiley & Sons.

Wieringa, R. (2014). Design Science Methodology for Information Systems and Software Engineering Vol. [Kindle DX version]. Retrieved from amazon.com

Yin, R. K. (2002). *Case study research: Design and methods* (Vol. 5): AGE Publications, Incorporated

Yoder, J. W., Balaguer, F., & Johnson, R. (2001). Architecture and design of adaptive object-models. *ACM Sigplan Notices, 36*(12), 50-60.

Zachman, J. A. (1987). A framework for information systems architecture. *Ibm Systems Journal, 26*(3), 276-292.

Zhang, H., Li, J., Zhu, L., Jeffery, R., Liu, Y., Wang, Q., & Li, M. (2014). Investigating dependencies in software requirements for change propagation analysis. *Information and Software Technology, 56*(1), 40-53.

Zickert, F., & Beck, R. (2012). Coping with Existing Systems in Information Systems Development. *IEEE Transactions on Software Engineering 38*(5), 1027-1039.

List of publications

Tarenskeen, D., Bakker, R., & Joosten, S. (2013). *Using ampersand in IT architecture*. Paper presented at the 7th International Conference on Research and Practical Issues of Enterprise Information Systems CONFENIS 2013, Prague, Czech Republic.

Tarenskeen, D., Bakker, R., & Joosten, S. (2015). Applying the V Model and Axiomatic design in the Domain of IT Architecture Practice. *Procedia CIRP*, *34*, 263-268.

Tarenskeen, D., Hoppenbrouwers, S., & van de Wetering, R. (2018). Reflections on Using an Architecture Model for Matching Existing Applications to a Radical Business Requirements Change: a Case Study. In *IFIP Working Conference on The Practice of Enterprise Modeling*, 383-393. Springer, Cham.

Tarenskeen, D. (2016, April). Conceptual Independence as an Architecture Pattern for Adaptable Systems. In *Proceedings of the 10th Travelling Conference on Pattern Languages of Programs,* 1-10. https://dl.acm.org/doi/10.1145/3022636.3022641

Tarenskeen, D., & Bakker, R. (2017). Applying Axiomatic design and Conceptual independence in the domain of IT systems. In *MATEC Web of Conferences,* 127(01006). EDP Sciences.

Tarenskeen, D., Van de Wetering, R., & Bakker, R. (2018). Unintended effects of dependencies in source code on the flexibility of IT in organizations. In *FedCSIS (Communication Papers)*, 87-94.

Tarenskeen, D., van de Wetering, R., Bakker, R., & Brinkkemper, S. (2020). The Contribution of Conceptual Independence to IT Infrastructure Flexibility: The Case of openEHR. *Health Policy and Technology, 9*(2), 233-244. doi: https://doi.org/10.1016/j.hlpt.2020.04.001

Tarenskeen, D., Van de Wetering, R., Bakker, R., & Brinkkemper, S. (2020). *Investigating the Impact of Outsourcing on IT Flexibility: The Conceptual Independence Perspective*. Submitted for publication.

Other publications

Tarenskeen, D., & De Vries, R. (2017). The mijnMáximaPlan, Hackathon Healthcare, Princess Máxima Center, A Case Study. *Producing an App for Supporting the Patient Journey at the Princess Máxima Center, Specialists in Pediatric Oncology* Retrieved 27-05-2020, The Open Group Publications, from https://publications.opengroup.org/y181

Summary

Healthcare organizations in the Netherlands are searching for ways to adapt the Information Technology (IT) infrastructure to new demands. These demands originate from external factors in society and technology. Healthcare organizations are, for instance, confronted with developments in healthcare itself and developments of technology in IT and medical hardware. In healthcare, medical processes are being changed, mainly because of patients' increasing involvement. Moreover, medical processes have to be adapted to the deployment of IT innovations, such as machine learning and the application of e-health systems. E-health systems that can be used independently of location will need connections to a diversity of other systems to exchange information. The challenges that organizations face lead to organization transformations, intensified management of the alignment of business and IT, and a broad spectrum of IT adaptations.

This research initially observed the approach of IT architects in managing required changes in the years after 2012. We have found that change in existing IT applications and information systems (IS) remains difficult for IT architects in healthcare because of numerous factors that need further study and explanation. After the first observational stage, we decided and elaborated on the concept of IT flexibility. Since the IT architects and healthcare organizations deal with a broad spectrum of requests, they have to prioritize requirements and eventually evaluate the degree of IT flexibility needed. We initially conformed to their interpretation of IT flexibility.

In the next phase, we examined the role that IT resources and other IT capabilities play in improving IT flexibility. The central research question in this study is formulated as: "Which IT resources and other IT capabilities contribute to IT flexibility in healthcare organizations in a changing environment in the Netherlands?"

By exploring the field, bottlenecks to IT flexibility were identified and discussed with IT architects. IT architects expressed problems adapting IT applications to new business and functional requirements. We describe IT applications as information systems for users in healthcare organizations. We focus on these applications as data-intensive IT applications, a specific form of IS. These IT applications are part of the IT infrastructure viewed as the set of configurations of interconnected systems and IT applications in the organization.

In the first research project, the Design science approach was selected. We observed the role models play in practice in changing the IT infrastructure in a direction desired by the organization. Cases in a large hospital, a regional hospital and a large home care facility revealed the complexity and size of the IT infrastructures in these organizations. The research resulted in a conceptual model for matching requirements to existing IT systems.

Evaluations of the conceptual model show that IT architects value the conceptual model, especially for managing complexity, but that healthcare organizations have difficulties in applying the recommendations of the IT architects afterwards. We noted that it is difficult to map new requirements to the existing IT infrastructure. The lack of an overview in the organization regarding functional requirements previously imposed on application components makes it difficult to implement changes in existing systems at the right place.

Relevant here is the design of modular systems, as described in Axiomatic design (AD). AD assumes that all requirements can be assigned to independent components in an existing IT infrastructure. The principle is called the Independence axiom (Suh, 1998). The axiom allows functional requirements to be changed independently of each other in the software. However, the extensive case studies of IT architectures showed that the independence axiom could not be found in the existing IT architectures. However, a dichotomy in IT architecture models is becoming clear. The first type of architecture model is a high-level model based on strategic goals, in which the details of technical IT components were missing. The second type of IT architecture model is designed to support technical management for IT operations and IT departments. The latter model is based on risk analysis and provides full technical details.

Concluding, the work of IT architects in practice suggests that software components are interconnected and difficult to disconnect. Information systems in particular were difficult to replace or adapt.

By consulting existing literature, we found a large number of articles that shed light on the inherent inflexibility of software. These software features hinder software adaptation, especially when conceptual models are concerned. Conceptual models in software development play a crucial role. They structure the concepts and terminology in such a way that the concepts are recognizable to users. They form the basis for the construction of data models for data-intensive software applications. Conceptual models are directly linked to users' understanding of the system and the functional requirements.

Our hypothesis that conceptual models play an important role in realizing IT flexibility based on functional requirements was prompted by McGinnes and Kapros (McGinnes & Kapros, 2015), who argue that underlying software structures can significantly improve or hinder the flexibility of IS. Their design principle of Conceptual Independence (CI) for IS recommends a design in software that separates the conceptual models from the application logic. The resulting IS operate on conceptual models that can be modified without having to reprogram the entire software application.

Analysis of existing open source software for Electronic Health Record (EPD) systems indicated that openEHR was a promising candidate to study the principle of CI.

In the last phase of the study, remarkable differences were found in organizations that had implemented openEHR in the IT infrastructure and organizations that did not use openEHR modules. This was demonstrated in a multi-case study conducted at 10 mental health organizations. It could be concluded from the results that it is precisely the specific characteristics of CI in openEHR that have significantly improved the flexibility of the entire IT infrastructure of the organization. The effect was not limited to the openEHR modules. The influence of an independent conceptual model in the IS was even observable in the relationship between IT supplier and organization, which influenced the IT outsourcing practice.

Finally, this Ph.D. study points to the influence of underlying software design on IT flexibility. The design of the IS as an IT resource in particular has a substantial effect on IT flexibility. We argue that the influence of the implementation of CI extends to the entire organization-wide IT infrastructure. We show that in the IT outsourcing practice of organizations that have implemented CI, the IT outsourcing relationship is also changing. This relationship strengthens IT flexibility. Moreover, this study opens an area of research in which software structure can be positioned as a factor in the context of organizational flexibility and agile organizations.

Samenvatting

Zorgorganisaties in Nederland zoeken naar manieren om de informatietechnologie (IT) -infrastructuur aan te passen aan nieuwe eisen. Deze eisen komen voort uit externe factoren in de samenleving en technologie. Zo worden zorgorganisaties geconfronteerd met ontwikkelingen in de zorg zelf en met technologische ontwikkelingen op het gebied van IT en medische hardware. In de zorg veranderen de medische processen, vooral door het beleid om patiënten steeds meer te betrekken bij het medische proces. Bovendien worden medische processen aangepast aan nieuwe technologie, zoals machine learning en de toepassing van e-healthsystemen. E-health-systemen, die plaats-onafhankelijk kunnen worden gebruikt, hebben verbindingen nodig met verschillende patiënt registratiesystemen om informatie uit te wisselen. De uitdagingen waar organisaties voor staan, leiden tot organisatietransformaties, intensiever beheer van de afstemming van business en IT, en een breed spectrum aan IT-aanpassingen.

We observeerden eerst de benadering van IT-architecten bij het managen van vereiste veranderingen in de jaren na 2012. We hebben geconstateerd dat het veranderen van bestaande systemen moeilijk blijft voor IT-architecten in de gezondheidszorg vanwege tal van factoren die nader onderzoek en uitleg behoeven. Na de eerste observatiefase hebben we een keuze gemaakt en hebben we het concept IT-flexibiliteit verder bestudeerd. Omdat de IT-architecten en zorgorganisaties te maken hebben met een breed spectrum aan verzoeken, moeten zij de prioriteiten stellen en uiteindelijk zelf de mate van IT-flexibiliteit bepalen die nodig is. Daarom hebben wij IT-flexibiliteit onderzocht vanuit het werk van de architecten.

In de volgende fase hebben we de rol onderzocht die IT-resources en andere IT-capabilities spelen bij het verbeteren van IT-flexibiliteit. De centrale onderzoeksvraag in dit onderzoek is geformuleerd als: "Welke IT-resources en andere IT-capabilities dragen bij aan IT-flexibiliteit bij zorgorganisaties in een veranderende omgeving in Nederland?"

In onderzoek naar de praktijk werden knelpunten voor IT-flexibiliteit waargenomen en besproken met IT-architecten. IT-architecten uitten problemen met het aanpassen van IT-applicaties aan nieuwe business eisen en functionele vereisten. Wij omschrijven IT-applicaties als informatiesystemen voor gebruikers in de zorgorganisaties. We richten ons op deze applicaties als data-intensieve IT-applicaties, een specifieke vorm van informatiesystemen (IS). Deze maken deel uit van de IT-infrastructuur als het geheel van configuraties van onderling gekoppelde systemen en IT-toepassingen in de organisatie.

In het eerste onderzoeksproject is de aanpak van Design science geselecteerd. Wij documenteerden de informatie die architecten gebruikten tijdens het veranderen van de IT-infrastructuur in een door de organisatie gewenste richting. Cases in een groot ziekenhuis, een regionaal ziekenhuis en een grote thuiszorginstelling onthulden de complexiteit en omvang van de IT-infrastructuren in deze organisaties. Het onderzoek resulteerde in een conceptueel model om eisen af te stemmen op bestaande IT-systemen.

Evaluaties van het conceptuele model laten zien dat IT-architecten het conceptuele model waarderen, vooral voor het managen van complexiteit, maar dat de zorgorganisaties achteraf moeite hebben om de aanbevelingen van de IT-architecten toe te passen. We merkten op dat het moeilijk is om nieuwe vereisten te mappen op de bestaande IT-infrastructuur. Gebrek aan overzicht in de organisatie betreffende functionele eisen, die eerder aan applicatiecomponenten gesteld zijn, maakt het moeilijk om op de juiste plaats wijzigingen door te voeren in bestaande systemen.

Relevant hierbij is de opzet van modulaire systemen, zoals is beschreven in Axiomatic design (AD). AD veronderstelt dat alle vereisten kunnen worden toegewezen aan onafhankelijke componenten in een bestaande IT-infrastructuur. Hierdoor kunnen functionele eisen onafhankelijk van elkaar in de software veranderd worden. Het principe wordt genoemd het Independence axiom (Suh, 1998). Echter, de uitgebreide casestudies van IT-architecturen toonden aan dat het onafhankelijkheidsaxioma niet kon worden teruggevonden in de bestaande IT-architecturen. Wel wordt een tweedeling in IT-architectuurmodellen duidelijk. Het eerste type architectuurmodel is een model op hoog niveau gebaseerd op strategische doelen, waarin de details van technische IT-componenten ontbraken. Het tweede type IT-architectuurmodel wordt ontworpen ter ondersteuning van technisch beheer voor IT-operaties en IT-afdelingen. Dit laatste model is gebaseerd op risicoanalyse en biedt volledige technische details.

Onze resultaten suggereren dat het werk van IT-architecten in de praktijk gehinderd wordt door softwarecomponenten, die onderling verbonden zijn en moeilijk te ontkoppelen. Met name inhoud van informatiesystemen was moeilijk te vervangen of aan te passen.

Door bestaande literatuur te raadplegen, hebben we een groot aantal artikelen gevonden dat licht werpt op de inherente inflexibiliteit van software. Deze softwarekenmerken belemmeren aanpassingen van software, vooral wanneer deze de conceptuele modellen betreffen. Conceptuele modellen in softwareontwikkeling hebben een cruciale rol. Zij structureren de concepten en terminologie zodanig dat de concepten herkenbaar zijn voor gebruikers. Zij vormen de basis voor de constructie van datamodellen voor data-intensieve softwareapplicaties. Conceptuele modellen zijn direct gekoppeld aan het begrip van gebruikers van het systeem en de functionele vereisten.

Onze hypothese dat conceptuele modellen een belangrijke rol spelen bij het realiseren van flexibiliteit van IT op basis van functionele vereisten werd mede ingegeven door McGinnes en Kapros (McGinnes & Kapros, 2015), die stellen dat onderliggende softwarestructuren de flexibiliteit van IS aanzienlijk kunnen verbeteren of belemmeren. Hun ontwerpprincipe van conceptuele onafhankelijkheid (Conceptual independence, CI) beveelt een ontwerp in software aan, dat de conceptuele modellen scheidt van de applicatielogica. De resulterende IS werken op conceptuele modellen die kunnen worden gewijzigd zonder de hele softwaretoepassing te hoeven herprogrammeren.

Uit analyse van bestaande opensource-software voor Electronic Health Record (EPD) -systemen, kwam naar voren dat openEHR een veelbelovende kandidaat was om het principe van CI mee te bestuderen.

In de laatste fase van het onderzoek werden opmerkelijke verschillen gevonden in organisaties, die openEHR hadden geïmplementeerd in de IT-infrastructuur en organisaties, die geen openEHR modules gebruikten. Dit bleek uit een multi-case study uitgevoerd bij 10 organisaties in de geestelijke gezondheidszorg. Uit de resultaten kon worden geconcludeerd, dat juist de specifieke kenmerken van CI in openEHR de flexibiliteit van de gehele IT-infrastructuur van de organisatie aanzienlijk hebben verbeterd. Het effect was niet beperkt tot de openEHR modulen. De invloed van een onafhankelijk conceptueel model in de IS was zelfs waarneembaar in de relatie tussen

IT-leverancier en organisatie, wat de IT-outsourcing praktijk beïnvloedde.

Tot slot, deze Ph.D. studie wijst naar de invloed van de onderliggende software-ontwerpen op IT-flexibiliteit. Vooral het design van de IS als IT-resource heeft substantiële effecten heeft op IT-flexibiliteit. Wij stellen dat de invloed van de implementatie van CI zich uitstrekt tot de gehele organisatie-brede IT-infrastructuur. We laten zien dat in de IT-outsourcing praktijk van organisaties die CI hebben geïmplementeerd, ook de IT-outsourcing relatie verandert. Deze relatie versterkt de IT-flexibiliteit. Bovendien, opent deze studie een onderzoeksgebied waarin softwarestructuur als factor kan worden gepositioneerd in de context van organisatorische flexibiliteit en agile organisaties.

Curriculum Vitae

Studies and degrees
The studies that I have finished all have to do with three domains of interest, Psychology, Economics and Artificial intelligence. First in 1981 a study of Social Psychology has been completed at Radboud University, Nijmegen (MSc). Then, to be able to teach in economics I have received my teaching certificate for secondary and higher education in 1988, named “Staathuishoudkunde en Statistiek, MO-B”. My attention has been drawn to artificial intelligence leading to the study of Knowledge engineering organized and certified by Middlesex University with CIBIT Utrecht (MSc) that was completed in 1994.

Work
In professional context the IT subjects have interested me from the 1980s on. In 1987 I started as employee for construction of examinations in economics at Cito. The main part of the work consisted of co-developing and designing the first databases including data models for educational and exam test questions that were introduced at Cito in 1987. Also, I have been a co-client for internal and external governmental projects for development of information systems for educational testing.

After Cito in 1998 I have been working as a consultant and developer of data quality software. The software supplier, Human inference, develops intelligent software for large database systems among others in telecom.

The last 20 years I have been teaching and lecturing in Universities for applied sciences, the teaching concerned the ins and outs of internet application development and java programming. Also I have been able to contribute as project leader to a Startup incubator for technology starters, one of the two predecessors of Utrecht Inc., in the years 2003 to 2006 at Hogeschool Utrecht.

At HAN University of Applied Sciences research of IT architecture in healthcare was performed in relation to the Ph.D. study. This study was funded by the HAN.

Changes in societal and technological requirements in healthcare need IT services that are location independent and communicate with large, existing patient administration systems. On different levels in organizations IT professionals work on changing the existing IT infrastructure. Enterprise architects work on high-level views of strategic goals and requirements, IT architects design solutions based on recent technology, compatible with older information systems. IT departments maintain and run current systems.

This study focuses on the IT practice of changing existing systems to new requirements. The high-level view on IT systems presupposes adaptability on a technical level. This presumption often proved to be false. Bottlenecks that surfaced in the research suggest that difficulties in applying the models were not caused by high-level inconsistencies or the methods and approaches applied but were positioned deep in software code. This finding has been a first major result in the study.

Next, new ideas emerged on the factors that could impede or aid flexibility. The principle of conceptual independence (CI) seems a promising approach. CI is based on a separation of conceptual models in software from the application logic, thereby enabling developers to change terminology or functionality in software without the need to reprogram the software application. The meta-standard of openEHR has been studied as a proxy for CI. The study showed that IT professionals do perceive an increase in IT flexibility when openEHR modules have been implemented. Next we found that the influence of flexible or inflexible software structures propagates through the IT systems and influences the agility of the organization when requirements change. This is a second major result. The conceptual model as used in practice needs space for evolution, also, in IT systems.

The author Debbie Tarenskeen has extensive experience in teaching and lecturing java programming for complex, distributed internet applications to Bachelor students.

Utrecht University

Open Universiteit

HAN_UNIVERSITY OF APPLIED SCIENCES