AN INVESTIGATION INTO THE IMPACT OF ENTERPRISE ARCHITECTURE DECISIONS ON THE RESPONSIBILITIES OF SOFTWARE DEVELOPERS IN COMPANIES THAT DEVELOP SOFTWARE

by

JUDITH VAN DER LINDE

submitted in part fulfilment of the degree of

MASTERS OF SCIENCE

in the subject

INFORMATION SYSTEMS

at the

UNIVERSITY OF SOUTH AFRICA

SUPERVISOR: PROF A J VAN DER MERWE

CO-SUPERVISOR: PROF A J GERBER

In Appreciation

“Creating a new theory is not like destroying an old barn and erecting a skyscraper in its place. It is rather like climbing a mountain, gaining new and wider views, discovering unexpected connections between our starting points and its rich environment. But the point from which we started out still exists and can be seen, although it appears smaller and forms a tiny part of our broad view gained by the mastery of the obstacles on our adventurous way up.” ~Albert Einstein

When this journey to complete the study started, it was like looking up at a mountain. Now, having reached the end of this journey, I can say that the journey up the mountain expanded my knowledge, gave me new views and expectations, and more than once threw a few surprises my way. It was a truly rewarding experience.

I would like to thank my patient and supportive supervisors, Prof Alta van der Merwe and Prof Aurona Geber. Without their joining in this *adventure*, it would never have been completed. Your mentoring, guidance, experience, ideas and life’s lessons inspired me.

Thank you to *The Company*, who, even though anonymous, has given me the opportunity to do research. Thank you to those colleagues who were willing to participate in the study. I appreciate your time and commitment to the study.

A special thank you to my brother, Wiehan, who was always willing to help, listen and to sometimes scold when it was necessary.

Finally, thank you to my friends and family, specifically my parents, Desere and Kobus, who shared many moments of joy and worry over the past years. I will always remember your support and encouragement during the challenging times and the many celebrations we shared after significant milestones were reached.

Abstract

Enterprise Architecture endeavours to resolve the complexity of increasingly distributed systems by aligning business vision with IT strategy, which in turn should reduce the overall costs of IT in the business and provide simpler, better and faster solutions to business problems. There are many Enterprise Architecture frameworks.

The main purpose of most of these frameworks is to assist with the challenges of managing the increased complexity of distributed systems, aligning business vision with IT strategy and reducing IT costs.

Many of the studies which produced the results stating Enterprise Architecture aligns business vision and reduces IT costs, were based on Zachman's work, and most of the published Enterprise Architecture success stories focus on the benefits provided to the company with regards to IT. In contrast very little documentation could be found that addresses the impact of Enterprise Architecture implementations on the individuals and systems within a company. If the individuals as the main implementers of any strategy are impacted negatively by Enterprise Architecture management decisions, there would be a negative impact on the return on investment of the company.

Enterprise Architecture allows the use of overlapping departments' processes and data, which translates into less development time as system components would already exist. Changes that are made to the Enterprise Architecture result in several additional changes that had to be implemented by the software developers. These changes influenced the workload, roles and responsibilities of the developers in such a way that the development team became negative about the additional work.

The purpose of this study was to investigate the impact of Enterprise Architecture management decisions on the responsibilities, work experience and attitude towards Enterprise Architecture of the software developers in a company that develops software by exploring and describing the nature of software development.

Based on the findings of this study, a list of impact of Enterprise Architecture decisions on the responsibilities of software developers in companies that develop software were identified. In this respect, the study identified impacts of Enterprise Architecture management decisions as well as possible solutions to these impacts.

Keywords

Enterprise Architecture, Software Developer, SDLC, TOGAF, Software development companies, Impact of Enterprise Architecture, Zachman, TOGAF ADM, Software developer roles and responsibilities

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Background

With the publication of an article *A Framework for Information Systems Architecture* in the IBM Systems Journal, John Zachman pioneered the field of Enterprise Architecture (Zachman, 1987). In this article, Zachman discussed the challenges and the vision of Enterprise Architecture. According to Zachman the main challenge of enterprises in the 21st century is the management of complexity and change in increasingly distributed systems:

> *The cost involved and the success of the business depends increasingly on its information systems and require a disciplined approach to the management of those systems (Zachman, 1987, p.276).*

Enterprise Architecture endeavours to resolve the complexity of increasingly distributed systems by aligning business vision with IT strategy, which in turn should reduce the overall costs of IT in the business and provide simpler, better and faster solutions to business problems (Zachman, 1987, p.276; Suomi et al., 2006, p.4; Ahlemann et al., 2012, p.10; Tomkowicz, 2007, p.10). To solve the challenges of managing the increased complexity of distributed systems, aligning business vision with IT strategy and reducing IT costs, Zachman (1987, p.276) stated that:

> *...it is necessary to use some logical construct (or architecture) for defining and controlling the interfaces and integration of all the components of the system.*

Zachman (1987, p.276) further suggested that in order to facilitate the management of the logical construct:

> *… it likely will be necessary to develop some kind of framework for rationalizing the various architectural concepts and specifications in order to provide for clarity of professional communication, to allow for improving and integrating development methodologies and tools, and to establish credibility and confidence in the investment of systems resources.*

This necessity to organise Enterprise Architecture led to the development of many Enterprise Architecture frameworks in the recent past. The main purpose of most of these frameworks is to assist with the challenges of managing the increased complexity of distributed systems, aligning business vision with IT strategy and reducing IT costs (Tambouris et al., 2011, p.150; Okunieff et al., 2011, p.58). There are many approaches for doing Enterprise Architecture (Tambouris et al., 2011, p.150; Bernard, 2012, p.109). For each of these multiple approaches there are frameworks, which are provided by different parties to serve different purposes (Tambouris et al., 2011, p.150; Okunieff et al., 2011, p.58).

An example of such an Enterprise Architecture framework is TOGAF, which has been in refinement since its creation as TAFIM in 1994 (The-Open-Group, 2009). TOGAF is at present an industry standard architecture framework (Van, 2006, p.21; The-Open-Group, 2009). It has been developed and improved since the mid-90's by IT professionals, working in The Open Group's Architecture Forum (The-Open-Group, 2009). TOGAF is a comprehensive method and set of supporting resources for Enterprise Architecture (The-Open-Group, 2009; Meaden and Whelan, 2012, p.204; Raynard, 2008, p.59). The latest version of TOGAF is Version 9. With the help of Enterprise Architecture frameworks such as TOGAF, companies are able to design, build and evaluate an Enterprise Architecture, which is appropriate for their company (Raynard, 2008, p.59).

As TOGAF is at present one of the most adopted and cited frameworks, it is plausible to argue that a company that wishes to implement Enterprise Architecture should benefit from the use of TOGAF (Raynard, 2008, p.60; The-Open-Group, 2009; Greefhorst and Proper, 2011, p.183). This was confirmed by an investigation done by the company used in the case study within the context of this research.

1.1.1 Contextualization and problem statement

As an introduction to the research conducted in this study, an overview is given of a typical context where the problem addressed in this research was noticed. The company where I as a participatory researcher worked operates within the financial sector. For the remainder of the dissertation the company used for contextualization will be referred to as *the Company.* The *Company* consists of multiple departments and each department functions as a separate business unit. One of these business units is the department that provides Group and IT Services, including the development of all the software systems that are used within the *Company*. The development of the software systems used in the *Company* requires multidisciplinary skills, one of which is software development. During the past decade the *Company* delivered a large number of these software systems. These systems were developed by coding the systems either from a new idea, or improving on the systems that already existed within the *Company*. The Group and IT Services department aims to keep up to date with the latest technology and incorporates system wide architecture changes when the technology drive requires it.

The *Company* caters for a niche market where the departments (or business units), which form part of the *Company*, supplement each other. The departments share data and processes that flow from one department to other departments. This sharing of data and processes resulted in such complexity that it influenced system development and generally the time spent on system development was too long. This delay in the delivery of software systems resulted in a low return on investment on IT systems and capabilities, as well as an increased turnaround time on projects. The delay in software project delivery resulted in a situation where the *Company* could not quickly respond to market trends, which reduced its competitiveness. This prompted an investigation by the *Company* executive committee on how to solve the issues of low return on investment and turnaround time on IT systems and capabilities.

The investigation the *Company* did on the implementation of an acceptable Enterprise Architecture framework resulted in a recommendation to implement an Enterprise Architecture solution, notably TOGAF, in the *Company*. The investigation suggested that TOGAF would decrease time to market, thus solving the problem of responding to market trends quickly (The-Open-Group, 2006, p.398; Raynard, 2008, p.33; Mohapatra and Singh, 2012, p.36). It also indicated how the Enterprise Architecture of the *Company* can use *overlapping* departments' processes and data to the advantage of the *Company*, translating into less development time as system components would already exist (Giachetti, 2010, p.110; Raynard, 2008, p.104). In summary, the implementation of TOGAF should provide the *Company* with solutions to the problems the *Company* was experiencing with regards to its IT and business alignment and system complexity (Lee, 2011, p. 65; Khosrow-Pour, 2006, p.541).

With the implementation of TOGAF, the *Company* integrated the System Development Life Cycle (SDLC) used by the Group and IT Services department into the Enterprise Architecture process. This was meant to enable the Enterprise Architecture management team to *manage* the system changes that needed to be made to adapt the software systems to the Enterprise Architecture. As a result of these system changes there was a substantial impact on the software developers of the *Company* with regards to their responsibilities and work experience, as well as in the end their attitude towards Enterprise Architecture and the influence Enterprise Architecture management decisions on their responsibilities.

For example, changes made to the *Company's* Enterprise Architecture resulted in several additional changes that had to be implemented by the software developers. These changes influenced the workload, roles and responsibilities of the developers in such a way that the development team became negative about the additional work. Software developers generally felt that the result of changes to the Enterprise Architecture could not be "as bad as it was", but it made their work extremely difficult.

Several of the software developers in the *Company* complained about issues and the difficulties they experienced as result of the Enterprise Architecture management decisions. The nature of these complaints further motivated me to achieve an improved understanding of the impact Enterprise Architecture management decisions have on the responsibilities and work experience of software developers. This impact would in turn influence the attitude of software developers towards the Enterprise Architecture management decisions in companies that develop software. There were many discussions on the impact of the introduction of Enterprise Architecture practices. Software developers were stating the negative impacts of Enterprise Architecture management level decisions on their responsibilities, work experience and attitude towards Enterprise Architecture. Examples of these statements include:

I wish the architects would think things through before saying something can be done. I will have to work three hours overtime tonight just to get the component out of the current system it's being used in – Software Developer.

This roll out was a big issue. Development had their hands full as the other system components failed when the software was released. They should really learn not to use components from other systems – Support Analyst

I wish my boss would listen to me when I say that it won't work. I told them last time we cannot just copy and paste code because we have that functionality in another system. – Senior Software Developer

The above statements give an indication that in this specific case, there were some issues experienced by software developers with the decisions made on an Enterprise Architecture management level. Furthermore, there were misconceptions with regards to the use of the software components within the development of the Enterprise Architecture. This experience contradicts the original view that Enterprise Architecture should provide solutions to the problems the *Company* was experiencing with regards to its IT and business alignment and system complexity (Lee, 2011, p. 65; Khosrow-Pour, 2006, p.541).

In literature, many studies were documented that investigate the solutions of the alignment and complexity challenges promised by the implementation of an Enterprise Architecture (Mahmood and Hill, 2011, p.12; Mykityshyn, 2007, p.84; Saha et al., 2009, p.265; Aiguier et al., 2010, p.45). A preliminary investigation into the literature revealed that many of the studies were based on Zachman's work, and that most of the published Enterprise Architecture success stories focus on the benefits provided to the company with regards to IT (Beker, 2011, p.245; Dan et al., 2010, p.45; Saha, 2007, p.161; Okunieff et al., 2011, p.17; Ahlemann et al., 2012). In contrast very little documentation could be found that addresses the impact of Enterprise Architecture implementations on the individuals and systems within a company. This lack of research on the impact of Enterprise Architecture on individuals could be regarded as an important deficiency in the area, especially if one considers that Enterprise Architecture promises benefits with regards to the return on investment in IT. It is possible to argue that if individuals as the main implementers of any strategy are impacted negatively by Enterprise Architecture management decisions, there would be an negative impact on the return on investment of the company (Bray, 2012, p.20; Desai, 2009, p.60; Langer, 2007, p.27). This argument provides the basis of the research conducted in this study, and the motivation and purpose of the study is discussed further in the next section.

1.2 Purpose of the study

The purpose of this study was to investigate the impact of Enterprise Architecture management decisions on the responsibilities, work experience and attitude towards Enterprise Architecture of the software developers in a company that develops software.

Goikoetxea claims that successful Enterprise Architecture is

> *… all about "lining up the ducks", so that the institutional side, the business side, the engineering side and the financial side of the picture are all addressed as a necessary condition to be met prior to and during the actual construction of the Enterprise Architecture (Goikoetxea, 2007, p.403)*

When we consider this quotation, there is little or no mention of the technological and software development aspects when addressing the engineering aspect of the Enterprise Architecture, specifically within a company that develops software. However, all changes that are made to the Enterprise Architecture have a ripple effect through the company and these changes will have an impact on the jobs and responsibilities of employees throughout the company (The-Open-Group, 2009, p.183; Hoque, 2002, p.87).

At present very few resources in Information Systems literature could be found that address the impact and challenges, which companies, which develop software, experience when they adopt Enterprise Architecture. The purpose of this study is to identify and investigate the impact of Enterprise Architecture management level decisions on employees, specifically software developers, with regards to the responsibilities, work experience and attitude towards Enterprise Architecture in companies that develop software.

1.3 Research Questions

During the implementation of a company's Enterprise Architecture numerous technical and organizational issues need to be addressed (Saha, 2009, p.27; Andersen et al., 2011, p.27). These challenges will naturally differ according to the environment or context of the Enterprise Architecture implementation. The purpose of this study was to investigate the impact of Enterprise Architecture decisions on the responsibilities, work experience and attitude towards Enterprise Architecture of software developers in companies that develop software.

The primary research question addressed by this study was:

> *How are the responsibilities, work experience and attitude of software developers impacted by the decisions made on an Enterprise Architecture management level in companies that develop software?*

During the research design, the study was divided into four sub questions. The sub questions were of an exploratory nature and served to differentiate and guide the study:

1. *According to literature, how do decisions made on an Enterprise Architecture management level impact the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer in companies that develop software?* The purpose of this question was to identify whether published literature and/or solutions to the study being done existed.
2. *How are the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer impacted by Enterprise Architecture management decisions?* The purpose of this question was to see from an observer point of view, what initial impact could be identified.
3. *How do software developers experience the impact of Enterprise Architecture management decisions on their responsibilities, work experience and attitude towards Enterprise Architecture?* The aim of this question was to obtain feedback from software developers on how they experience the impact of Enterprise Architecture management decisions on their responsibilities.

4. *What are the solutions that could address the impact of Enterprise Architecture management decisions on software developer's responsibilities?* This aim of this research question was to incorporate feedback and possible solutions from the software developers.

These sub-research questions were addressed by using the research strategy described in the following section.

1.4 Research Strategy

Since the main focus of a research study is to discover answers that will help explain and achieve the aim and objectives of the planned research, a methodological approach is required to facilitate the research process (Maykut, 1994, p.43; Mitchell and Jolley, 2009, p.53; Munizzo and Musial, 2010, p. 30). The strategy adopted in this study was an interpretive qualitative ethnographic case study, which was supplemented by surveys (Cohen et al., 2007, p.255; Wiebe et al., 2009, p. 597; Simons, 2009, p. 22; Lee et al., 1997, p. 278).

The initial observation of a problem was made from informal discussions with fellow colleagues. This was followed by a primary list of impacts that was gathered through the use of participant observation. During this exercise the researcher gained insight through conversations with other software developers in the case study environment (Spindler and Hammond, 2006, p.34; DeWalt and DeWalt, 2010, p.1; Angrosino, 2008, p.4; Murchison, 2010, p.7). After the initial lists of issues were compiled, a detailed literature study was executed to determine whether literature addresses the impact on employees of a company that executes an Enterprise Architecture implementation. This study provided the theoretical underpinning, which enabled the researcher to establish a thorough background on Enterprise Architecture, TOGAF, Software development and the Software Development Life Cycle (SDLC). The theoretical framework provided the contextual background for the study as well as the basis for the analysis of the findings of the study (Cassell and Symon, 2004, p. 324; Farquhar, 2012, p. 37; Runeson et al., 2012; Wiebe et al., 2009, p.813). The study also focused on the specific responsibilities of software developers within the phases of the SDLC, which is included in the theoretical framework of this study. Because of the lack of literature on research with regards to the impact of Enterprise Architecture management decisions on the responsibilities, work experience and attitude towards Enterprise Architecture of software developers in companies that develop software during the initial investigation, this study was motivated and executed.

After the literature review was conducted, the list of initial impacts identified through researcher participant observation was structured into interview questions. The interviews were conducted as semi-structured field interviews using open-ended questions (Klandermans and Staggenborg, 2002, p.93; Flick, 2009, p.165; Remenyi, 2011, p.20). The interview questions were designed to allow the participants to agree or disagree with the findings of the participant observation, as well as to give room in the study for the experiences of other software developers. These interviews also helped to determine the attitude of the software developers towards Enterprise Architecture. The software developers were also asked to comment on the impacts of Enterprise Architecture management decisions on their responsibilities.

The impacts obtained from these interviews were compared to the initial list, which was obtained from the participant observation, and impacts were either confirmed or updated with new information. In some instances, the descriptions of the impact were altered to generalize or group the different impacts accordingly (Wiebe et al., 2009, p.474; Outhwaite and Turner, 2007, p. 107).

This list of impacts was converted into a survey comprising of yes/no questions. The survey asked software developers whether they agreed with a specific impact or not. This survey was distributed to software developers in other companies who adopted Enterprise Architecture. The purpose of the survey was to confirm the ethnographic case study results. In addition to the confirmation of the results, the survey allowed the researcher to uncover different facets to the study and enabled an investigation through an appropriate combination of methods and sources (Marschan-Piekkari and Welch, 2004, p.129; Brown, 2008, p. 221).

This confirmed list of impacts and possible solutions were compiled and formalized as the contribution of the study. The next section discusses the context, scope and limitations, which were applicable during the study.

1.5 Context, Scope and Limitations

This study defines a list of impacts and possible solutions to the impact of Enterprise Architecture management decisions on the responsibilities, work experience and attitude towards Enterprise Architecture of software developers in companies that develop software. The *Company* that provides the context for this study is a company comprising of departments. One of these departments specializes in developing software for the *Company*, which in turn provides financial solutions to a niche market. Because the context is important in Information Systems research, the context is discussed fully in Chapter 5.

The scope of this study was to investigate the impact of Enterprise Architecture management decisions on the responsibilities, work experience and attitude towards Enterprise Architecture of software developers in companies that develop software, specifically the *Company,* which was used as the case study. This study was restricted to the implementation of TOGAF, and does not consider or investigate the use of alternative Enterprise Architecture frameworks and approaches.

A software development team consists of software developers, testers, business analysts and business owners (Sobh and Elleithy, 2010, p.31; Leffingwell, 2010, p.34; Brennan, 2009, p. 10). This study is purposely limited to the impact of Enterprise Architecture management decisions in a software development company on the responsibilities, work experience and attitude towards Enterprise Architecture of *software developers* only, although impacts may be experienced by other employees and members of the team. Given sufficient time and resources, the study could be expanded to study the impacts on the full team, as well as a change in team dynamics.

1.6 Outline of Chapters

This dissertation comprises of 7 chapters. Figure 1 outlines the structure of the dissertation. At the beginning of each chapter the dissertation map will indicate in a green colour, the stage of the dissertation. A map of the specific chapter structure will follow after the dissertation chapter map.

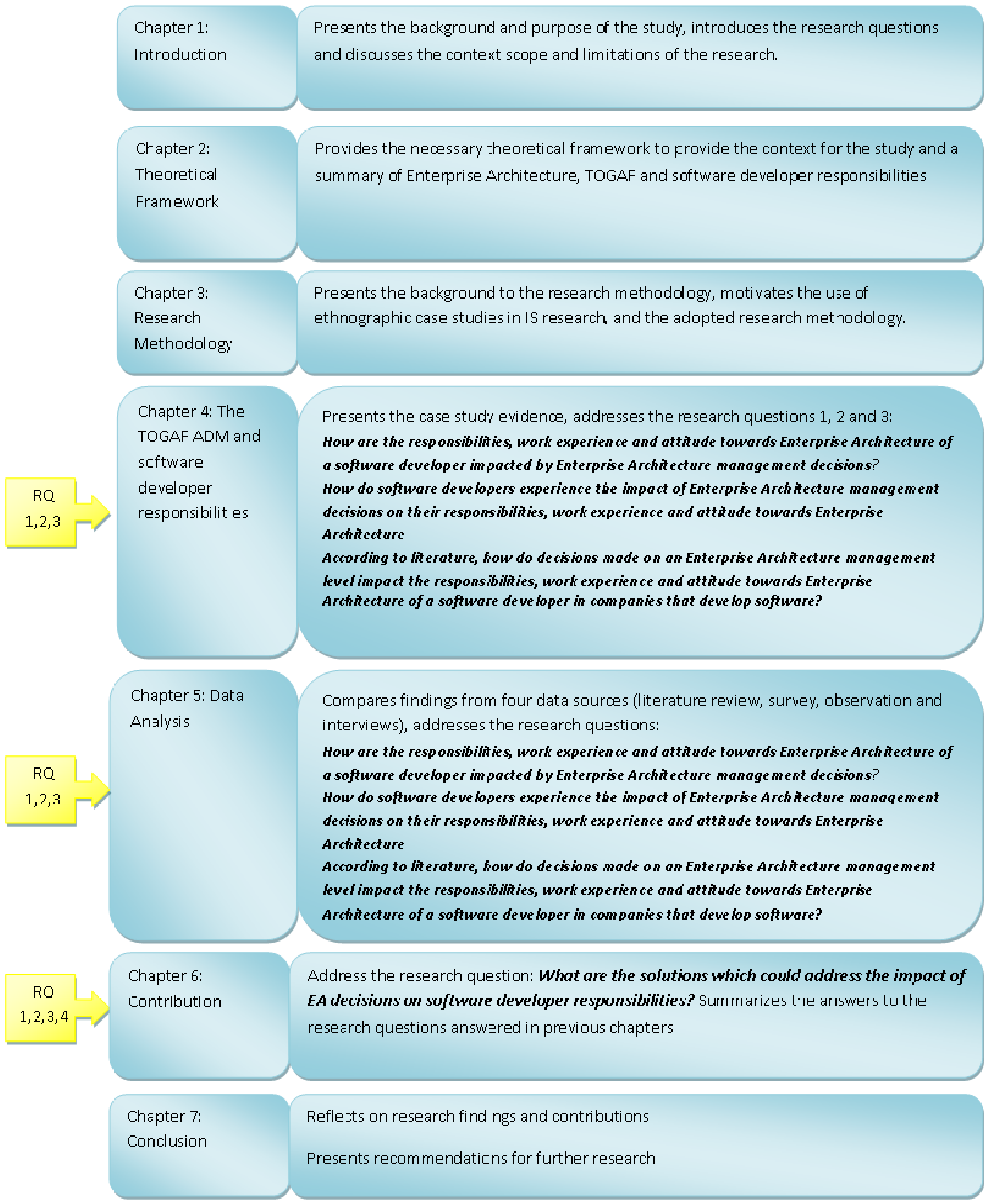
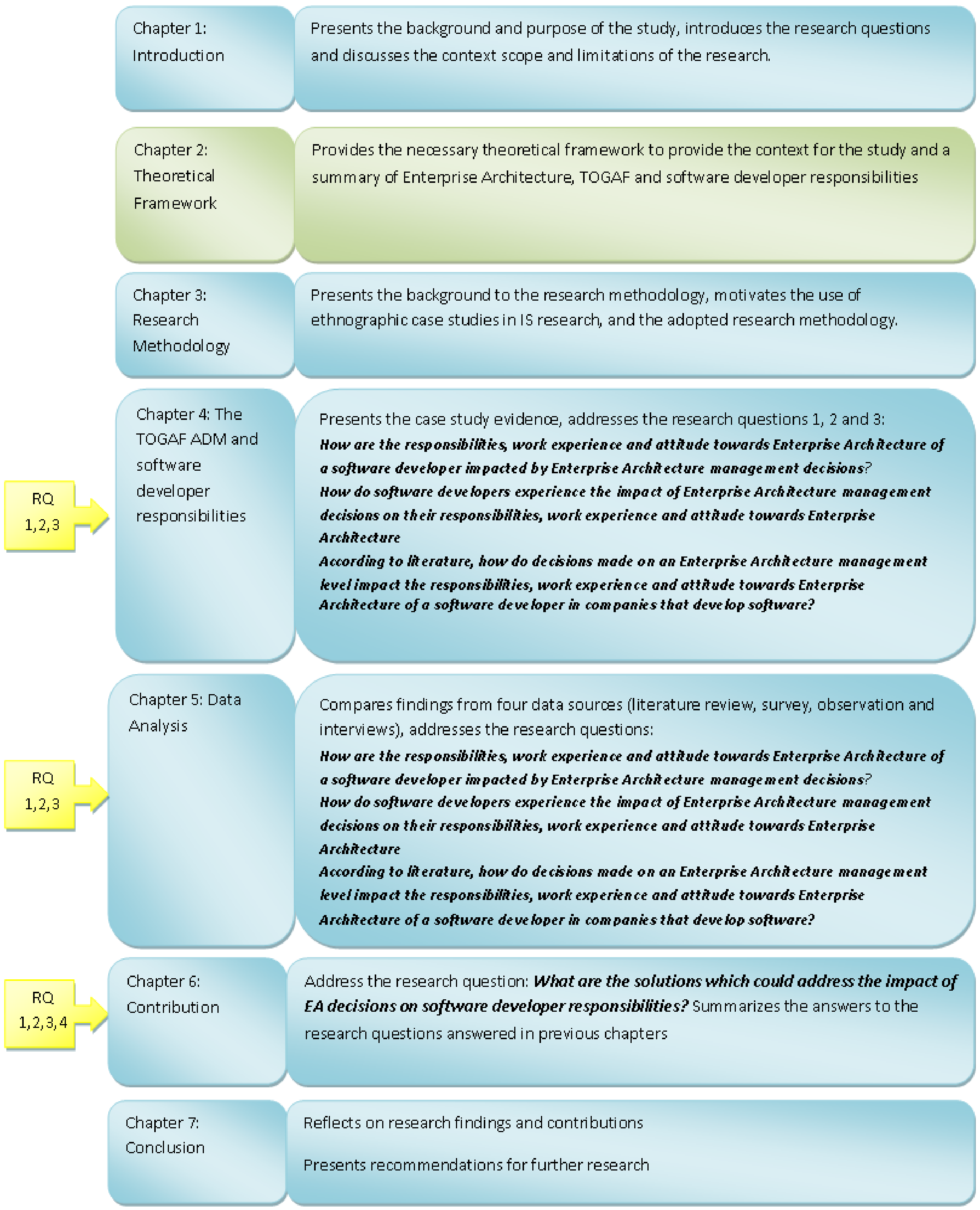

Chapter 1: Introduction
Presents the background and purpose of the study, introduces the research questions and discusses the context scope and limitations of the research.
Chapter 2: Theoretical Framework
Provides the necessary theoretical framework to provide the context for the study and a summary of Enterprise Architecture, TOGAF and software developer responsibilities
Chapter 3: Research Methodology
Presents the background to the research methodology, motivates the use of ethnographic case studies in IS research, and the adopted research methodology.
RQ 1,2,3
Chapter 4: The TOGAF ADM and software developer responsibilities
Presents the case study evidence, addresses the research questions 1, 2 and 3:
How are the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer impacted by Enterprise Architecture management decisions?
How do software developers experience the impact of Enterprise Architecture management decisions on their responsibilities, work experience and attitude towards Enterprise Architecture
According to literature, how do decisions made on an Enterprise Architecture management level impact the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer in companies that develop software?
RQ 1,2,3
Chapter 5: Data Analysis
Compares findings from four data sources (literature review, survey, observation and interviews), addresses the research questions:
How are the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer impacted by Enterprise Architecture management decisions?
How do software developers experience the impact of Enterprise Architecture management decisions on their responsibilities, work experience and attitude towards Enterprise Architecture
According to literature, how do decisions made on an Enterprise Architecture management level impact the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer in companies that develop software?
RQ 1,2,3,4
Chapter 6: Contribution
Address the research question: What are the solutions which could address the impact of EA decisions on software developer responsibilities? Summarizes the answers to the research questions answered in previous chapters
Chapter 7: Conclusion
Reflects on research findings and contributions
Presents recommendations for further research

Figure 1 Dissertation Chapter Map

Chapter 2: Theoretical Framework



Chapter 1: Introduction

Presents the background and purpose of the study, introduces the research questions and discusses the context scope and limitations of the research.

Chapter 2: Theoretical Framework

Provides the necessary theoretical framework to provide the context for the study and a summary of Enterprise Architecture, TOGAF and software developer responsibilities

Chapter 3: Research Methodology

Presents the background to the research methodology, motivates the use of ethnographic case studies in IS research, and the adopted research methodology.

RQ 1,2,3

Chapter 4: The TOGAF ADM and software developer responsibilities

Presents the case study evidence, addresses the research questions 1, 2 and 3:
How are the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer impacted by Enterprise Architecture management decisions?
How do software developers experience the impact of Enterprise Architecture management decisions on their responsibilities, work experience and attitude towards Enterprise Architecture
According to literature, how do decisions made on an Enterprise Architecture management level impact the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer in companies that develop software?

RQ 1,2,3

Chapter 5: Data Analysis

Compares findings from four data sources (literature review, survey, observation and interviews), addresses the research questions:
How are the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer impacted by Enterprise Architecture management decisions?
How do software developers experience the impact of Enterprise Architecture management decisions on their responsibilities, work experience and attitude towards Enterprise Architecture
According to literature, how do decisions made on an Enterprise Architecture management level impact the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer in companies that develop software?

RQ 1,2,3,4

Chapter 6: Contribution

Address the research question: ***What are the solutions which could address the impact of EA decisions on software developer responsibilities?*** Summarizes the answers to the research questions answered in previous chapters

Chapter 7: Conclusion

Reflects on research findings and contributions

Presents recommendations for further research

Figure 2 Chapter 2 Dissertation Chapter Map

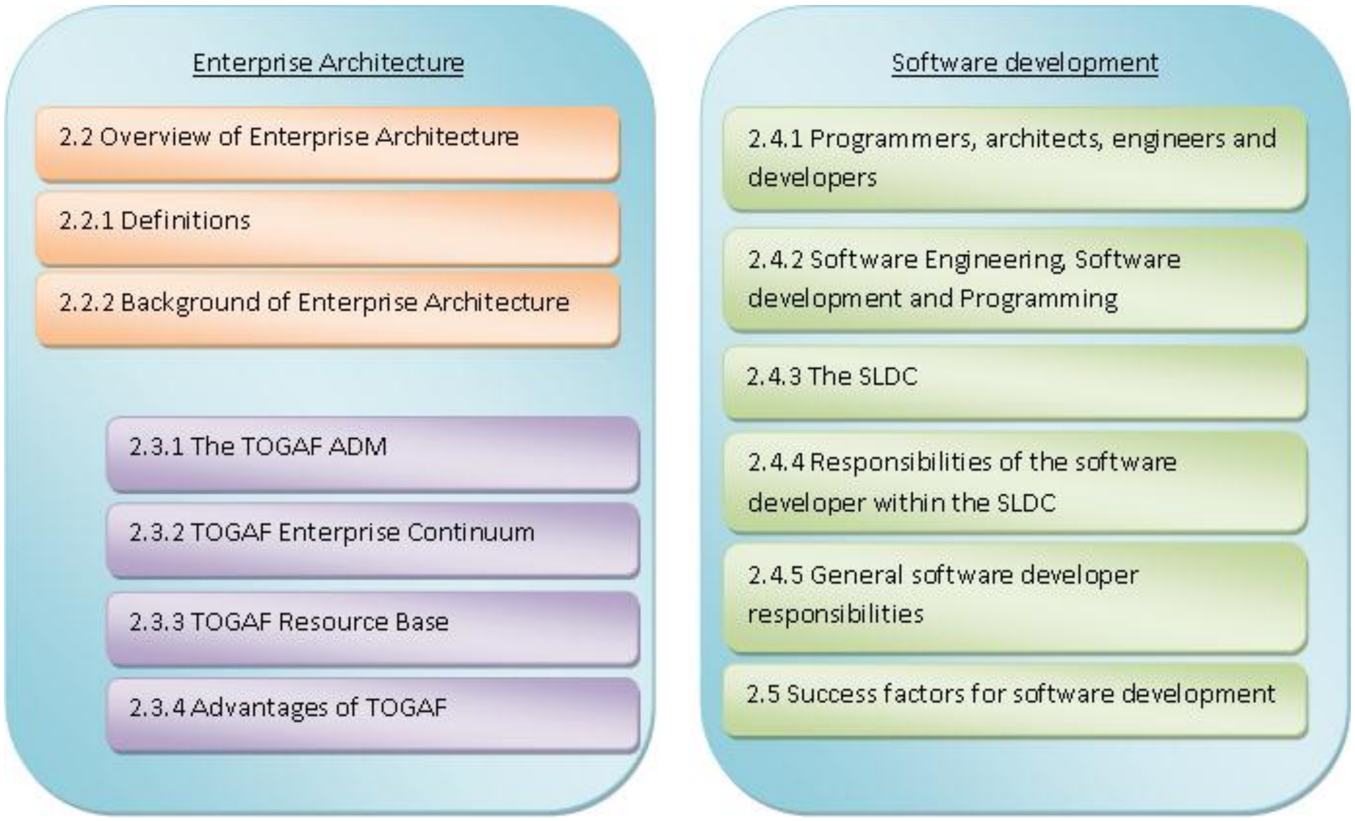

Enterprise Architecture
2.2 Overview of Enterprise Architecture
2.2.1 Definitions
2.2.2 Background of Enterprise Architecture
2.3.1 The TOGAF ADM
2.3.2 TOGAF Enterprise Continuum
2.3.3 TOGAF Resource Base
2.3.4 Advantages of TOGAF
Software development
2.4.1 Programmers, architects, engineers and developers
2.4.2 Software Engineering, Software development and Programming
2.4.3 The SLDC
2.4.4 Responsibilities of the software developer within the SLDC
2.4.5 General software developer responsibilities
2.5 Success factors for software development

Figure 3 Chapter 2 Chapter Map

2.1 Introduction

The literature survey firstly discusses the literature with regards to Enterprise Architecture. Enterprise Architecture and the development of the field associated with Enterprise Architecture are well documented. Specifically, several sources discuss the evolution of Enterprise Architecture frameworks such as TOGAF(Moller and Chaudhry, 2012, p.17). The aim of this chapter is to examine the history of Enterprise Architecture in an attempt to discover the context within which it stands (Moller and Chaudhry, 2012, p.17). This context is explored in terms of the diversity of definitions for Enterprise Architecture and the background of Enterprise Architecture. Because the context of this study includes an implementation of TOGAF specifically, a more in-depth examination is done on TOGAF and how the Architecture Development Method (ADM) is used to develop Enterprise Architecture. The TOGAF Enterprise Continuum and Resource base is used in conjunction with the ADM as an Enterprise Architecture Framework, which promises advantages to using TOGAF. This study of TOGAF provides context for the further investigation of how Enterprise Architecture is implemented and the *expected* results of an Enterprise Architecture adoption within a company.

As a secondary topic of this literature survey, software development as a related aspect is discussed. Because Enterprise Architecture and subsequent changes to the Enterprise Architecture might affect changes to the software systems being used in the company, software development and the software developers developing the software will be impacted (Giachetti, 2010, p.114; Greefhorst and Proper, 2011, p.125; Abramowicz et al., 2010, p.145). To understand and evaluate these impacts, a thorough understanding of who software developers are, the roles they fulfil, the System Development Life Cycle (SDLC), in which they have to work, the responsibilities assigned to software developer in the spectrum of the SDLC, as well as responsibilities that are not governed by the SLDC, is necessary.

After a preliminary search at the beginning of this study it was found that there are many sources on the *impact* of factors on Enterprise Architecture decisions, but limited literature exists on the how decisions made at Enterprise Architecture level impact the individuals and systems within an enterprise. This could be regarded as an important deficiency, especially if one considers that Enterprise Architecture promises benefits with regards to the return of investment in IT, which could not be realized without the involvement of individuals and employees in a company.

The remainder of this chapter provides an overview of Enterprise Architecture, as well as architecture frameworks, specifically TOGAF and the TOGAF ADM. In addition, it provides information on software development and the success factors of software development. This overview brings context to how the research questions are viewed and consequently addressed in later chapters (Cassell and Symon, 2004, p. 324; Farquhar, 2012, p. 37; Runeson et al., 2012; Wiebe et al., 2009, p. 813).

2.2 Overview of Enterprise Architecture

Enterprise Architecture (EA) is a relative recent phenomenon, which aims to address two prevalent problems in enterprises or companies, namely:

- System complexity—Companies were spending more money building IT systems (Sessions, 2007, p.1 ; Zachman, 1987, p.276; Tupper, 2011, p.26).
- Poor business alignment— Companies were finding it increasingly difficult to keep expensive IT systems aligned with business needs (Sessions, 2007, p.1; Zachman, 1987, p.276; Tupper, 2011, p.26).

Zachman (1987) is considered to be a pioneer of the field of Enterprise Architecture because of his suggestion for a *Framework for Information Systems Architecture* (Goikoetxea, 2007, p.335; Hammami et al., 2012, p. 319; Ahlemann et al., 2012, p. 207; Global and Staff, 2010). Zachman's view was that a holistic approach is required for the management of information system architecture. Such an approach would consider every relevant issue from all necessary perspectives. The framework Zachman developed over time was later renamed as *Zachman's Framework for Enterprise Architecture* (Tupper, 2011, p.26; Jaap Schekkerman, 2006, p. 131; Moller and Chaudhry, 2012, p. 20). The coining of the term *Enterprise Architecture* is attributed to Zachman (Saha, 2007, p.xx; Unhelkar, 2010, p. 447). After the initial definition of Zachman, several studies and discussions addresses the topic of Enterprise Architecture, and several new definitions emerged, which is evidence of the complexity of the field of Enterprise Architecture (Dane, 2010, p. 22). Therefore, the following section defines how Enterprise Architecture is contextualized in this study by providing a definition for Enterprise Architecture adopted for this study.

2.2.1 Definitions

According to the Oxford Online dictionary a *definition* is:

> *a statement of the exact meaning of a word or the nature or scope of something (Dictionary, 2012).*

To formulate a definition of Enterprise Architecture, it is necessary to discuss definitions of the composite terms *enterprise* and *architecture*.

2.2.1.1 Definition of Enterprise

An enterprise is "a business or company" (Oxford, 2009). According to The Open Group (2006) an *enterprise* is any collection of organizations that has a common set of goals and/or a single bottom line. Thus an enterprise can be a government agency, a whole corporation, a division of a corporation, a single department, or a chain of geographically distant organizations linked together by common ownership (Campbell, 2007).

2.2.1.2 Definition of Architecture

Campbell states architecture is "an abstraction or design of a *system*, its structure, components and how they interrelate or a family of guidelines (concepts, policies, principles, rules, patterns, interfaces and standards) to use when building a new IT capability" (Campbell, 2007).

The ANSI/IEEE STD 1472-200 defines architecture as:

> *The fundamental organization of a system, embodied in its components, their relationship to each other and the environment, and the principles governing its design and evolution (Laar and Punter, p.105, Halpin et al., 2009, p.175).*

Given the above definitions of *enterprise* and *architecture*, the next section discusses the composite term *Enterprise Architecture*.

2.2.1.3 Different definitions of Enterprise Architecture

There are multiple approaches for doing Enterprise Architecture and as result, there are different perspectives on Enterprise Architecture (Tambouris et al., 2011, p.150; Bernard, 2012, p.109). Therefore, current literature offers many different definitions of the term Enterprise Architecture, (Saha, 2007, p.147; Kiyoki, 2006, p.220; Lapalme, 2011). Some of the most prevalent definitions obtained in literature are listed below:

- "Enterprise Architecture is a strategic information asset base, which defines the mission, the information necessary to perform the mission and the technologies necessary to perform the mission, and the transitional processes for implementing new technologies in response to the changing mission needs(Remenyi, 2006, p.84)*."*
- "Enterprise Architecture is the set of descriptive representations relevant for describing an enterprise such that it can be produced to management's

requirements and maintained over its useful life (Camarinha-Matos et al., 2010, p.25)."

- "Enterprise Architecture is a complete expression of the enterprise; a master plan that acts as a collaboration force between aspects of business planning such as goals visions, strategies and governance principles; aspects of business operations such as business terms, organization structures, processes and data; aspects of automation such as information systems and databases and the enabling technological infrastructure of the business such as computers, operating systems and networks (Jaap Schekkerman, 2006, p.13)."

When scrutinizing these definitions, it is possible to argue that many of the available definitions for Enterprise Architecture are variations of the definition by Lapalme namely: *a description (and/or the process of achieving a description) of the interrelated components of an enterprise in order to guide their evolution* (Lapalme, 2011, p.2). The definition by Lapalme is therefore adopted as the definition for Enterprise Architecture in this study.

When considering the other definitions for Enterprise Architecture, they differ mainly with regards to two aspects namely scope and purpose (Lapalme, 2011, p.2). The first is the scope of the term *Enterprise a*nd the other aspect is *purpose* (Lapalme, 2011, p.2). Lapalme categorized Enterprise Architecture approaches according to scope and purpose into three major beliefs (Lapalme, 2011, p.2). These beliefs are summarized in the table below:

Table 1 Scopes and Purposes of the definitions of Enterprise Architecture adapted from (Lapalme, 2011, p.2)

Scope	***Purpose***
Enterprise wide IT platform (EIT) -all components (software, hardware, etc.) of the enterprise IT assets.	Effective enterprise strategy execution and operation through IT-Business alignment. The purpose is to enhance business strategy execution and operations. The primary means to this end is the aligning of the business and IT strategies so that the proper IT capabilities are developed to support current and future

	business needs.
Enterprise (E) - the enterprise as a socio-cultural—techno-economic system; hence all the facets of the enterprise are considered – the enterprise IT assets being one facet.	Effective enterprise strategy implementation through execution coherency. The purpose is effective enterprise strategy implement. The primary means to this end is designing the various facets of the enterprise (governance structures, IT capabilities, remuneration policies, work design, etc.) to maximize coherency between them and minimize contradictions.
Enterprise-in-Environment (EiE) - Includes the enterprise scope but adds the environment of the enterprise as a key component as well as the relationships and transactions between the enterprise and its environment.	Innovation and adaption through organizational learning. The purpose is organizational innovation and adaption.

The *Company* which forms the context of this study endeavoured to implement Enterprise Architecture adopting an enterprise wide IT platform scope with effective enterprise strategy execution and operation through IT-Business alignment. This belief provides the context and viewpoint of the participant researcher and thus influences the way the research questions are addressed in later chapters.

Having defined Enterprise Architecture, as well as the purpose with regards to Enterprise Architecture of the *Company*, the next section provides background to how Enterprise Architecture started and how the field developed.

2.2.2 Background of Enterprise Architecture

With the publication of an article *A Framework for Information Systems Architecture* in the IBM Systems Journal, John Zachman pioneered the field of Enterprise Architecture (Zachman, 1987). In this paper Zachman articulated the challenges and the vision of Enterprise Architecture. He claimed that the challenge faced by enterprises is to manage the complexity of increasingly distributed systems:

The cost involved and the success of the business depends increasingly on its information systems and require a disciplined approach to the management of those systems (Zachman, 1987, p.276).

Enterprise Architecture promises to resolve this challenge by aligning business vision with IT strategy, which will reduce the overall costs of IT in the business and provide simpler, better and faster solutions to business problems (Zachman, 1987, p.276; Suomi et al., 2006, p.4; Ahlemann et al., 2012, p.10; Tomkowicz, 2007, p.10). To solve the challenges of managing the increased complexity of distributed systems, aligning business vision with IT strategy and reducing IT costs, Zachman (1987, p.276) stated that:

...it is necessary to use some logical construct (or architecture) for defining and controlling the interfaces and integration of all the components of the system.

Zachman (1987, p.276) suggested that in order to facilitate the management of the logical construct:

… it likely will be necessary to develop some kind of framework for rationalizing the various architectural concepts and specifications in order to provide for clarity of professional communication, to allow for improving and integrating development methodologies and tools, and to establish credibility and confidence in the investment of systems resources.

The need to organise Enterprise Architecture aspects, components and approaches led to the development of many Enterprise Architecture frameworks. These frameworks generally aims to assist with the challenges of managing the increased complexity of distributed systems, aligning business vision with IT strategy and reducing IT costs (Tambouris et al., 2011, p.150; Okunieff et al., 2011, p.58).

Another argument that supports Enterprise Architecture adoption is the growing costs of the information systems that are necessary for the success of the company. To contain these costs, a disciplined approach to the management of those systems is required (Zachman, 1987, p.276; Tupper, 2011, p26; Varajao et al., 2010, p.55).

Zachman was a key influence on one of the earliest ventures to create Enterprise Architecture (Ahlemann et al., 2012, p.208). The venture was implemented by a branch of the U.S. Government in the Department of Defence. TAFIM, known as the Technical Architecture Framework for Information Management, was introduced in 1994 (Goikoetxea, 2007, p.30; Kahin and Abbate, 1995, p.541). TAFIM was noticed by the U.S. Congress (Ahlemann et al., 2012, p.208; Hammami et al., 2012, p.35; Kahin and Abbate, 1995, p.554). Because of the promised benefits of TAFIM, Congress in 1996 passed a bill known as the Clinger-Cohen Act of 1996 (Information Technology Management Reform Act), which stipulated that all federal agencies take steps to improve the effectiveness of their IT investments. A CIO Council, consisting of CIO's from all major governmental bodies, was created to oversee this effort (Goikoetxea, 2007, p.25; Jaap Schekkerman, 2006, p.57; Ahlemann et al., 2012, p.208).

In April 1998, the CIO Council began work on its first major project, the Federal Enterprise Architecture Framework (FEAF) (Saha, 2007, p. 3; Bernard, 2012, p. 273; Hammami et al., 2012, p.319). Version 1.1 of this framework was released in September of 1999.This framework contained some new ideas, one being segmented architectures (The-Open-Group, 2006, p.26; Lillehagen and Krogstie, 2008, p.94). This meant Enterprise Architecture could focus on segmented subsets of the larger company (Goikoetxea, 2007, p.29; Okunieff et al., 2011, p. 74).

Over time, responsibility for federal Enterprise Architecture moved from the CIO Council to the Office of Management and Budget (OMB). In 2002, the OMB evolved and renamed the FEAF methodology as the Federal Enterprise Architecture (FEA) (Goikoetxea, 2007, p.34; Scholl, 2010, p.288; Bernard, 2012, p.273; Green et al., 2010, p.66). At this stage the work done on TAFIM was turned over to The Open Group. They turned it into a new standard that is currently known as The Open Group Architecture Framework (TOGAF) (Goikoetxea, 2007, p.259; Ahlemann et al., 2012, p.208; Raynard, 2008, p.21; The-Open-Group, 2007). TOGAF is discussed in more detail in Section 2.3.

In 2005, when OMB was becoming the dominant Enterprise Architecture force in the public sector, another organization was taking steps to become a dominant force in the private sector. This group was Gartner (Bente et al., 2012, p.119).

By 2005, Gartner was already one of the most influential organizations specializing in CIO-level consulting. However, in the specific area of Enterprise Architecture, the best known IT research and advisory group was not Gartner, but Meta Group (Bente et al., 2012, p.119). Gartner had tried to build an enterprise-architecture practice, but never reached the status of the Meta Group. In 2005, Gartner decided that because they couldn't compete with Meta Group they would buy it (Sessions, 2007, p.1; Bente et al., 2012, p.119).

After the purchase of Meta Group, Gartner/Meta took a year to appraise what each company was able to contribute as far as Enterprise Architecture experience and methodologies (Bente et al., 2012, p.119).

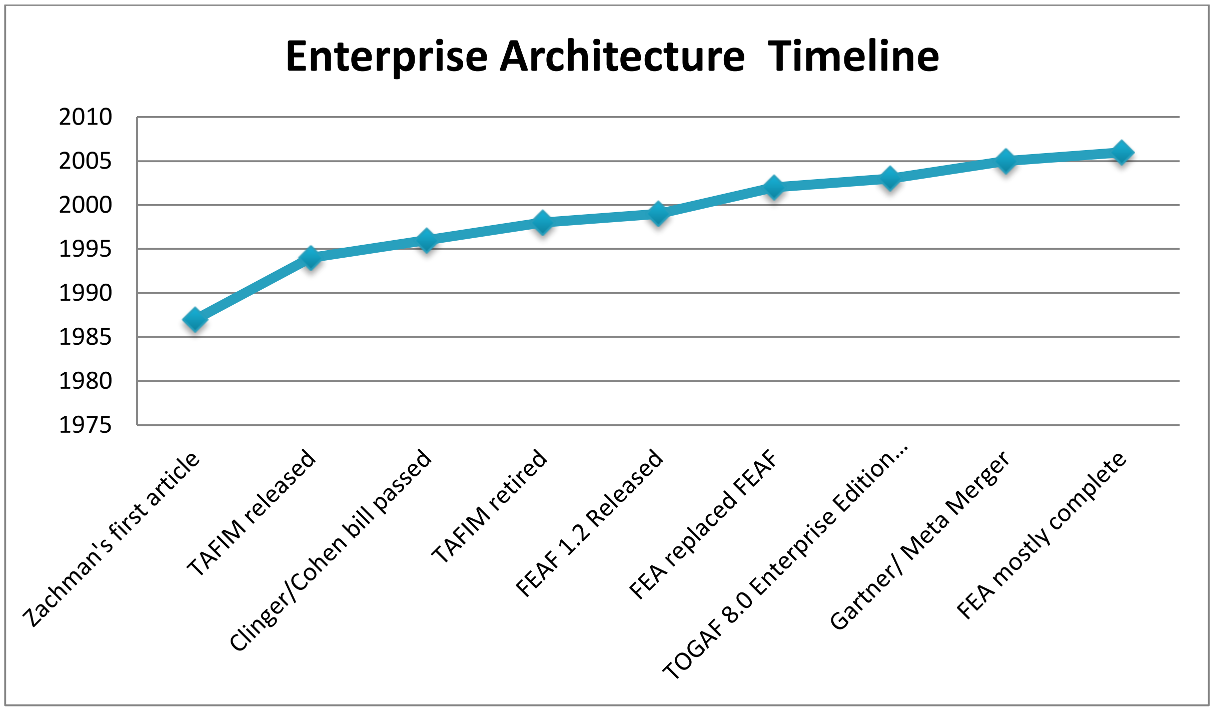


Figure 4 Enterprise Architecture Timeline

This graph summarizes this history with an enterprise-architecture timeline. The next section discusses Enterprise Architecture maturity, as it is one of the metrics used to identify how successful Enterprise Architecture is in the company.

2.2.3 Enterprise Architecture Maturity

Enterprise Architecture (EA) is a relative recent phenomenon, which aims to address two prevalent problems in enterprises or companies, namely:

- System complexity—Companies were spending more money building IT systems (Sessions, 2007, p.1; Zachman, 1987, p.276; Tupper, 2011, p.26).
- Poor business alignment— Companies were finding it increasingly difficult to keep expensive IT systems aligned with business needs (Sessions, 2007, p.1; Zachman, 1987, p.276; Tupper, 2011, p.26).

Many organizations are still engaged in developing and implementing fully mature Enterprise Architecture(Filip et al., 2008, p.240).To assess the successfulness of Enterprise Architecture, maturity is used as a metric to identify the success of an Enterprise Architecture Implementation. Ross et al. presented a two-dimensional operating model that depicts levels of Enterprise Architecture maturity. The more mature the company, the better the benefits promised with the use of Enterprise Architecture. This model is comprised of four quadrants that represent different combinations of the levels of business integration and standardization. Table 2 presents the two dimensional operating model:

Table 2 Characteristics of the four operating models adapted from (Ross et al., 2006)

<u>Coordination:</u>	**<u>Unification</u>**:
• Shared Customers • Shared Products • Shared Suppliers • Impact on other business unit transactions • Operationally unique business units or functions • Autonomous business management • Business unit control over business unit processes and	• Customers or supplies may be local or global • Globally integrated business processes often with support of enterprise systems • Business unite with similar or overlapping operations • Centralized management often applying functional/processes/ business unit matrices • High level process owners design standardized processes

business process design • Shared data • Consensus processes for designing IT infrastructure services: IT application decisions made in business units	• Centrally mandated databases • IT decisions made centrally
Diversification: • Few, if any, shared customers or suppliers • Independent transactions • Operationally unique business units • Autonomous business management • Business unit control over business unit design • Few data standards across business units • Most IT decisions made within business units	**Replication**: • Few, if any shared customers • Independent transaction aggregated at a high level • Operationally similar business units • Autonomous business unit leaders with limited discretion over processes • Centralized control over business process design • Standardized data definitions but data locally owned with some aggregation at enterprise • Centrally mandated IT services

The goal is to work from a diversification model to a unification model to align business vision with IT strategy, reducing the overall costs of IT in the business and providing simpler, better and faster solutions to business problems. Because Enterprise Architecture mainly exists as models of the company and its processes, documentation plays an important role in Enterprise Architecture maturity.

Enterprise Architecture documentation is one of the measurement factors for Enterprise Architecture maturity (Hanschke, 2010,p. 194). Estimating maturity in Enterprise Architecture management requires an appraisal of content, processes, organization, steering and tool support(Ross et al., 2006, p.47, Hanschke, 2010, p.194). Estimating maturity in Enterprise Architecture management requires an appraisal of content, processes, organization, steering and tool support(Ross et al., 2006, p.47; Hanschke, 2010, p.194). The following aspects are important:

- Completeness – have all models and the interactions between them been documented? Have all parts of the enterprise been documented or just some parts of the enterprise?
- Granularity, up-to-datedness, quality and consistency and
- Ease of maintenance.

Because TOGAF is the Enterprise Architecture adopted in the case study, which is addressed in Chapter 5, a more in-depth study on TOGAF is presented in the next section.

2.3 TOGAF

TOGAF, which was derived from TAFIM, was initiated in January 2000 and has remained an open source architecture framework (Goikoetxea, 2007, p.34; Hausman and Cook, 2010, p.26; Raynard, 2008, p.21; Perks and Beveridge, 2002, p.79). The TOGAF Enterprise Architecture Framework consists of four main components namely:

- TOGAF ADM;
- TOGAF Enterprise Continuum;
    - TOGAF Foundation Architecture;
    - Integrated information Infrastructure Model;
- TOGAF Resource Base.

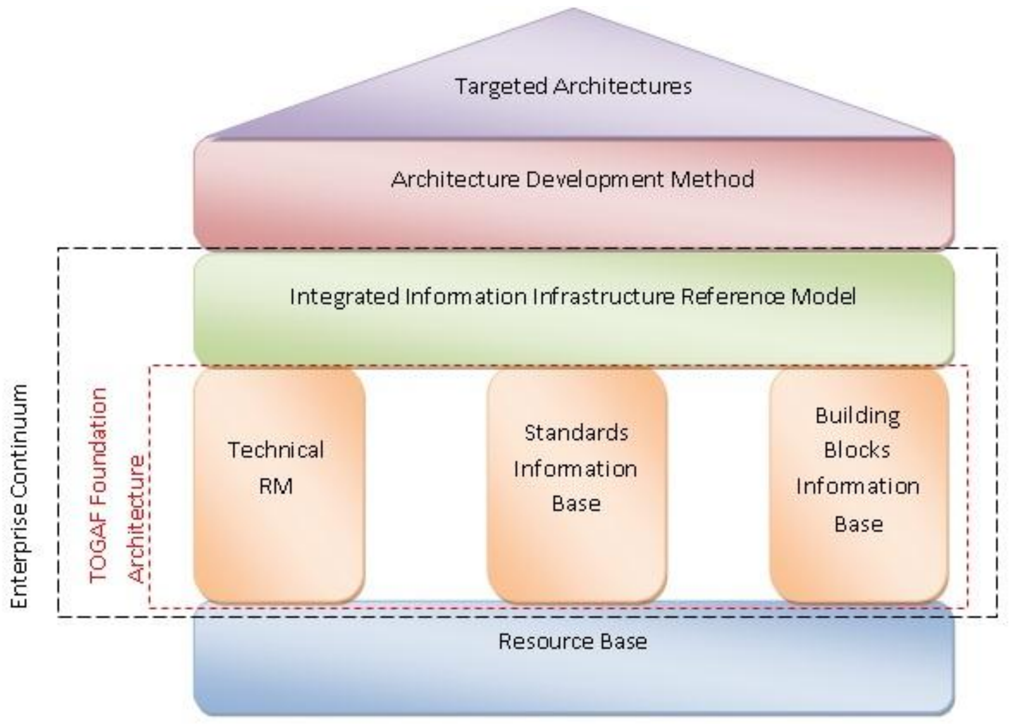

Targeted Architectures
Architecture Development Method
Integrated Information Infrastructure Reference Model
Technical RM
Standards Information Base
Building Blocks Information Base
Resource Base
Enterprise Continuum
TOGAF Foundation Architecture

Figure 5 TOGAF Based On (Lankhorst, 2005, p.26)

TOGAF divides Enterprise Architecture into four categories, as follows:

- Business architecture—Describes the processes the business uses to meet its goals (Tupper, 2011, p.29; Ziemann, 2011, p.79);
- Application architecture—Describes how specific applications are designed and how they interact with each other (Tupper, 2011, p.29; Ziemann, 2011, p.79);
- Data architecture—Describes how the enterprise data stores are organized and accessed (Tupper, 2011, p.29; Ziemann, 2011, p.79); as well as
- Technical architecture—Describes the hardware and software infrastructure that supports applications and their interactions (Tupper, 2011, p.29; Ziemann, 2011, p.79) .

The following sections discuss the TOGAF ADM, the Enterprise Continuum and the TOGAF Resource Base.

2.3.1 The TOGAF ADM

The TOGAF ADM describes a method for developing Enterprise Architecture, and forms the core of TOGAF. In this section the TOGAF ADM is discussed in more detail because of its close alignment with the development of the systems in the case study context.

TOGAF is not a technology or tool specific framework (Lankhorst, 2005, p.25, Blevins et al., 2007, p.111). TOGAF can be used for developing the products used by any of the other frameworks – such as the Zachman Framework, Federal Enterprise Architecture Framework (FEAF), Treasury Enterprise Architecture Framework (TEAF), and C4ISR/DoD Framework. TOGAF can be used by any type of deliverable that the enterprise uses (Blevins et al., 2007; Raynard, 2008, p. 47; Perks and Beveridge, 2002, p.126; Khosrow-Pour, 2006, p.543).

The most prevalent part of TOGAF that represents the TOGAF approach is the Architecture Development Method, better known as the ADM, which is depicted on Figure 6. The ADM describes a technique for creating an enterprise architecture (Doom, 2010, p.43, Hass, 2007, p.57).

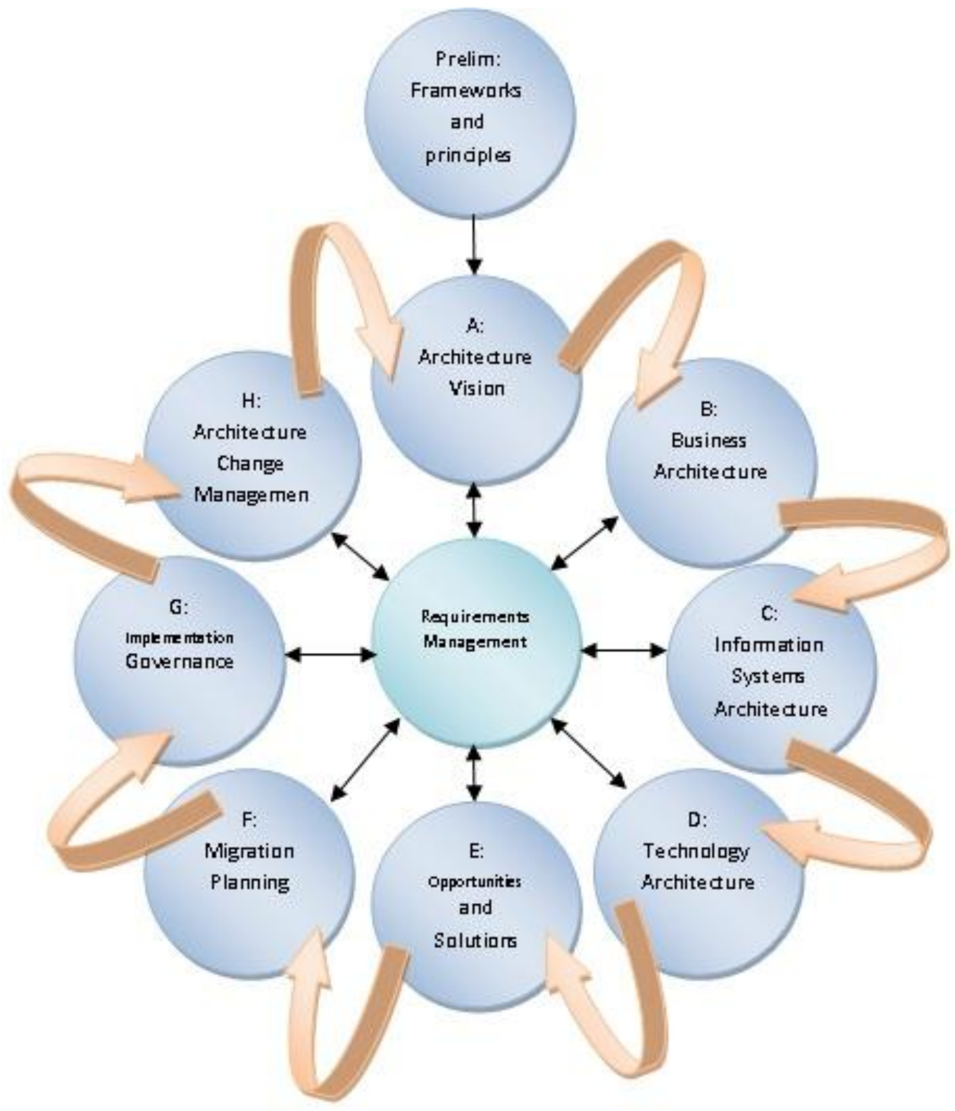

Prelim: Frameworks and principles
A: Architecture Vision
B: Business Architecture
C: Information Systems Architecture
D: Technology Architecture
E: Opportunities and Solutions
F: Migration Planning
G: Implementation Governance
H: Architecture Change Managemen
Requirements Management

Figure 6 TOGAF ADM Based On (Blevins et al., 2007, p.25)

The ADM proposes a cycle that allows the architect to view the Enterprise Architecture from different viewpoints, thus ensuring a full view of a complex structure (Blevins et al., 2007, p.62; Hass, 2007, p.57). The application of the ADM is an iterative process that consists of either repeating the full cycle from A-H, or comprising mini-cycles within the bigger cycle. During these iterations there should be constant checking to see if the current architecture is still in line with the original expectations. This validation must include the artefacts from the previous iteration, the scope, detail, schedules and milestones (Vasudeva, 2009, p. 20; Doom, 2010, p.43; Khosrow-Pour, 2006, p.543).

The activities in each phase of the ADM could include:

Table 3 Activities per phase in the ADM, adapted from (Blevins et al., 2007, p.26)

ADM Phase	***Activity***
Preliminary Phase: Frameworks and Principles	The organization must be prepared with the TOGAF procedures in order to facilitate a successful Enterprise Architecture project.
Requirements Management	The main expectation of this project is kept here. Each cycle or phase should be validated to the original expectations. Each of the phases store, prioritize, dispose or address requirements, the history is also kept here.
Phase A: Architecture Vision	The scope, constraints and expectations for the Enterprise Architecture project is setup and defined here. The business context is validated and the Statement of Architecture Work is created.
Phase B: Business Architecture Phase C: Information Systems Architecture (Data and Applications) Phase D: Technology Architecture	The Enterprise Architecture is developed on the three levels: • Business • Information Systems • Technology This process involves creating a current view of the existing architecture ("as-is") and the target architecture ("to-be"). This is then followed by a gap analysis to determine what changes should be made to the existing architecture.
Phase E: Opportunities and Solutions	When all the gaps have been identified and prioritized, these gaps are sorted into implementation plans.
Phase F: Migration Planning	A cost benefit analysis is done on all changes to be made. An Implementation Road Map is also created noting the analyses and prioritization of the requirements

Phase G: Implementation Governance	Implementation Architecture Contracts are created to ensure all work is compliant with the current architecture and that work that is carried out conforms to the new architecture.
Phase H: Architecture Change Management	Because business's vision changes, the architecture must change to accommodate the change in business. This phase ensures concurrency in the architecture

TOGAF proponents claim that business and IT alignment should be the result when using the ADM to develop Enterprise Architecture, as well as following all TOGAF guidelines (Saha, 2007, p.171; Bernard, 2012, p.77).

2.3.2 TOGAF Enterprise Continuum

An Enterprise Continuum (EC) stores a process model as well as the taxonomies of Enterprise Architecture-related building blocks. This includes standards, solutions, services, patterns, frameworks, and the technologies that can be used in conjunction with the ADM (van Sante and Van Den Bent, 2007, p.21; Okunieff et al., 2011).

TOGAF recommends a rendering of both the present and future architectures, as well as the transitional states, as a series of modelling artefacts and accompanying specification documents (Vasudeva, 2009, p.20; Raynard, 2008, p.184; Dubey, 2011, p.77; van Sante and Van Den Bent, 2007, p.17). These models may be from different architectural domains and differ in granularity and purpose.

In order to address the ambiguity that surrounds the business-to-technology divide, TOGAF has adopted an approach that breaks the Enterprise Architecture implementation and specification process into two separate but dependent model layers and three potentially different work streams that deal with architecture requirements, business agreements, and architectural solutions respectively (Rittgen, 2007, p.76; Lankhorst, 2005, p.73; Okunieff et al., 2011, p.75). The streams represent a natural divide for most companies where both enterprise planning and solution implementation groups exist:

- The Architecture Continuum (AC) stream offers a framework to analyze Enterprise Architecture both in context and scope, defining business agreements and classifying reusable assets according to Application Building Blocks (ABBs) (Raynard, 2008, p.88; Greefhorst and Proper, 2011, p.63; Vasudeva, 2009, p.20).
- The Solutions Continuum (SC) stream contributes a way to describe an implementation of the AC states with Solution Building Blocks (SBB) (van Sante and Van Den Bent, 2007, p.32; Perks and Beveridge, 2002, p.441; Vasudeva, 2009, p.20).

2.3.3 TOGAF Resource Base

The Open Group (2007) defines the TOGAF resource base as:

> *The TOGAF resource base is a set of resources – guidelines, templates, checklists, and other detailed materials – that support the TOGAF ADM.*

When the TOGAF ADM is applied in conjunction with the enterprise continuum and the TOGAF resource base, there are advantages for the company. These advantages are discussed in the next section.

2.3.4 Advantages of TOGAF

There are a variety of frameworks to choose from when implementing Enterprise Architecture. However, a comparison of the advantages and disadvantages of all the frameworks are out of scope for this study. The advantages and disadvantages of TOGAF are discussed because it is the framework that was adopted in the case study.

Advantages of TOGAF include:

- Enhanced profitability or reduced costs (Ross et al., 2006, p.93-100; Mahmood and Hill, 2011, p.36; Land et al., 2008, p.40);
- Quicker time to market (Ross et al., 2006, p.93-100; Mahmood and Hill, 2011, p.33; Land et al., 2008, p.40);
- Improved strategy execution (Ross et al., 2006, p.93-100; Land et al., 2008, p.40; Mahmood and Hill, 2011, p.32);
- Assistance with the analysis of alternate architectures (Land et al., 2008, p.41);

- Improved communication between stakeholders (Land et al., 2008, p.41);
- A full and coherent understanding of the enterprise (Land et al., 2008, p.42; Mahmood and Hill, 2011, p.34);
- A compass and atlas for management (Land et al., 2008, p.42);
- Business process improvement by structuring the business services as they are required (Land et al., 2008, p.42);
- Eliminating enterprise duplication, enabling the company to move to a shared services model (Land et al., 2008, p.42; Goikoetxea, 2007, p.23);
- Strategy translated into executable projects (Land et al., 2008, p.42);
- More reliability and security, and less risk (Mahmood and Hill, 2011, p.35; Goikoetxea, 2007, p.23; Ross et al., 2006, p.93-100); as well as
- Faster systems development and less complexity (Mahmood and Hill, 2011, p.37; Goikoetxea, 2007, p.23).

This is not a comprehensive list of all the advantages of TOGAF, but the list highlights the advantages specifically stated when TOGAF was chosen by the *Company* in the case study.

2.4 Software development

In this section the focus is on literature discussing aspects of software development, the development life cycle and responsibilities of software developers. These topics are important since this study investigated the impact of Enterprise Architecture management decisions on the responsibilities of software developers. It is therefore necessary to provide background on software development to contextualize the findings discussed in Chapter 5.

A definition of software development as phrased by Dooley is provided by the next quotation:

> *Software development is the process of taking a set of instructions from a user, analyzing them, designing a solution to the problem and then implementing that solution on a computer (Dooley, 2011, p.1).*

As can be seen from the definition by Dooley, software development is primarily concerned with the tasks of addressing user requirements by building a software system that meets the user requirements. There are many viewpoints as to who develop software and the different job titles applied to software developers. The following section discusses these job titles in the context of the case study.

2.4.1 Programmers, architects, engineers and developers

In this section we discuss the different job titles for software developers to contextualize the responsibilities of the software developers as seen in the case study. This also helps to differentiate between the responsibilities, which are allocated per job title. The following list mentions some of the job titles that may be applied to a software developer:

- *Software architects* are responsible for creating and maintaining the overall structure and layout of a software system's components and their interfaces within and outside of the system (Gutbrod and Wiele, 2012, p.22; Qian et al., 2009, p.18).
- A *systems engineer* (system architect, systems analyst) analyses the role of the system in the broader enterprise, defines the requirements the system needs to meet, in terms of services and non-functional requirements, and defines the architecture of the system to meet the requirements (Kossiakoff et al., 2010, p.80; Grady, 2006, p.143; Jamshidi, 2011).
- The *database developer* (database analysts, data modellers, or data architects) is responsible for leading the coordination and collection of database requirements, documenting, organizing and communicating the requirements for the database, modelling the database architecture and ensuring it supports the business needs (Fisher, 2004, p.29; Erbschloe, 2003, p.38).
- *Programmers* are responsible for developing and modifying programs (systems) to satisfy user requirements (Zak, 2010, Puntambekar, 2008, p.35; Stair and Reynolds, 2011, p.357; Perry, 2002, p.383).

The job titles discussed above are used to discuss the responsibilities or areas of expertise in relation to the type of job. However, it is often the case that software developers fulfil all of the duties mentioned above and the term *software developer* is used as a general term. Thus, for the purposes of this study, programmers, software developers, software architects, system engineers and database developers will all be referred to as *software developers*. However, for clarity about the different responsibilities with regards to software engineering, software development and programming, it is described in more detail in the next section.

2.4.2 Software Engineering, Software development and Programming

Depending on the job title applied to a software developer, certain responsibilities are associated with the job title. These job titles are classified in sections of software engineering, software development and programming. If a hierarchical structure could be applied to the concepts of software development, software engineering would be at the top of the structure. Software engineering is a process that includes software development, scheduling and estimation, project management, resource management, configuration management and baseline building (Dooley, 2011, p.1; Saleh, 2009, p.2; Mouratidis and Giorgini, 2006, p.2; Sangeeta, 2008, p.5).

Software development is an activity that forms part of software engineering (Dooley, 2011, p.1). Software development focuses on specific areas in the SDLC, which includes Design and Build, Coding (Programming) and Testing and Deployment. Programming is an activity that forms part of software development, which entails the writing of a computer program in a language understood by computers (Dooley, 2011, p.1; Kirikova, 2002, p.241; Kan, 2003, p.56).

Software development narrows the focus of software engineering, but broadens the scope of programming to include analysis, design and deployment (Dooley, 2011, p.1; Abrahamsson et al., 2009, p.187; Puntambekar, 2009, p.33). Figure 7 shows a graphical representation of the focus levels of software engineering, software development and programming:

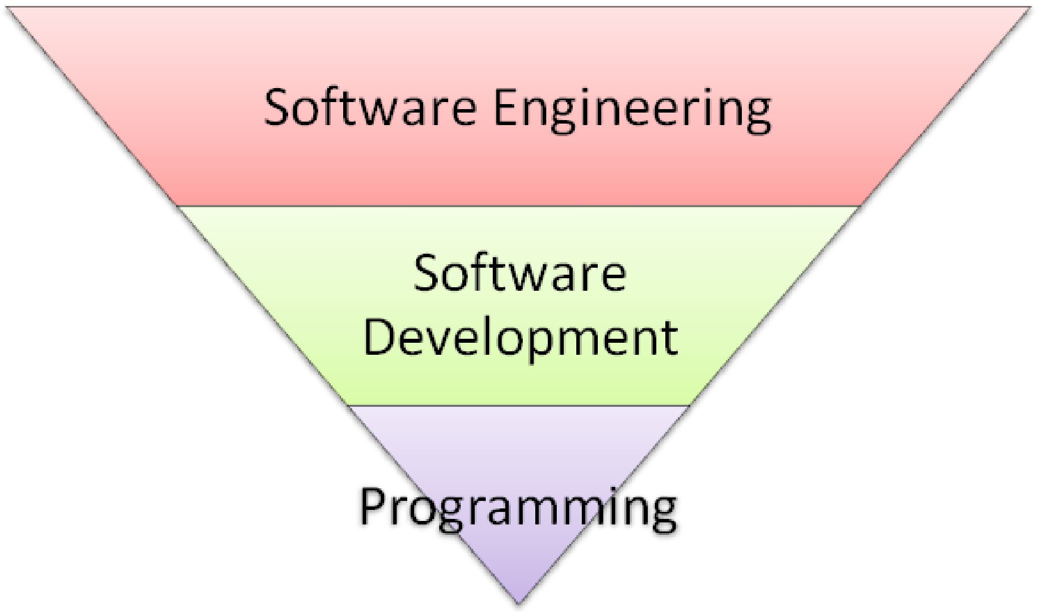


Figure 7 Hierarchical Structure of Software Engineering, Software Development and Programming

The figure is an upside down triangle with Software Engineering at the top level, because software engineering encompasses the full spectrum of activities included in the Information Systems field. Software development is a subset of software engineering and has narrower scope that software engineering, but encompasses more than programming which is at the bottom of the triangle and has the narrowest focus.

In the next section the System Development Life Cycle (SDLC) is discussed because the SDLC is the process that governs the software development process (Vellani, 2007, p.140; Mittra, 2002, p.5; O'Connor et al., 2011, p.85).

2.4.3 The SLDC

The SDLC is a set of models that software developers use as a pattern in developing computer systems (Lewis, 2008, p.16; Stair and Reynolds, 2011, p.428; Gill, 2011, p.147). There are numerous variations of the SDLC, and many companies may choose to customize one or all the stages in the SDLC to fit to their specific environment (Lewis, 2008, p.16; Gill, 2011, p.147).

The basic phases of the SDLC include:

- Analysis

- Design
- Build
- Test
- Deploy
- Maintain

There are two main basic SDLC models (Lewis, 2008, p.77; Satzinger et al., 2008, p.78):

- The single version model
- Iterative model

Single version models are models in which all phases are planned and tasked and each phase must be completed before the following phase is started (Lewis, 2008, p.77, Oates, 2006, p.39, Hausman and Cook, 2010, p.165, Satzinger et al., 2008, p.40). Examples of single version models include the Waterfall model, the V model and the Overlapping waterfall model.

Iterative models on the other hand include models where subsections of the software are passed through the phases as they are completed (Lewis, 2008, p.77; Oates, 2006, p.39; Hausman and Cook, 2010, p.165; Satzinger et al., 2008, p.40). Examples of iterative models include the Spiral and Fountain models.

The most prevalent form of the SDLC is the waterfall model (Figure 8 below) that represents that the phases mentioned above are executed in that order with no iterations, feedback loops or repetition of phases. When using this model each phase must be completed before moving on to the next phase (Lewis, 2008, p.77; Oates, 2006, p.39; Hausman and Cook, 2010, p.165; Satzinger et al., 2008, p.40).

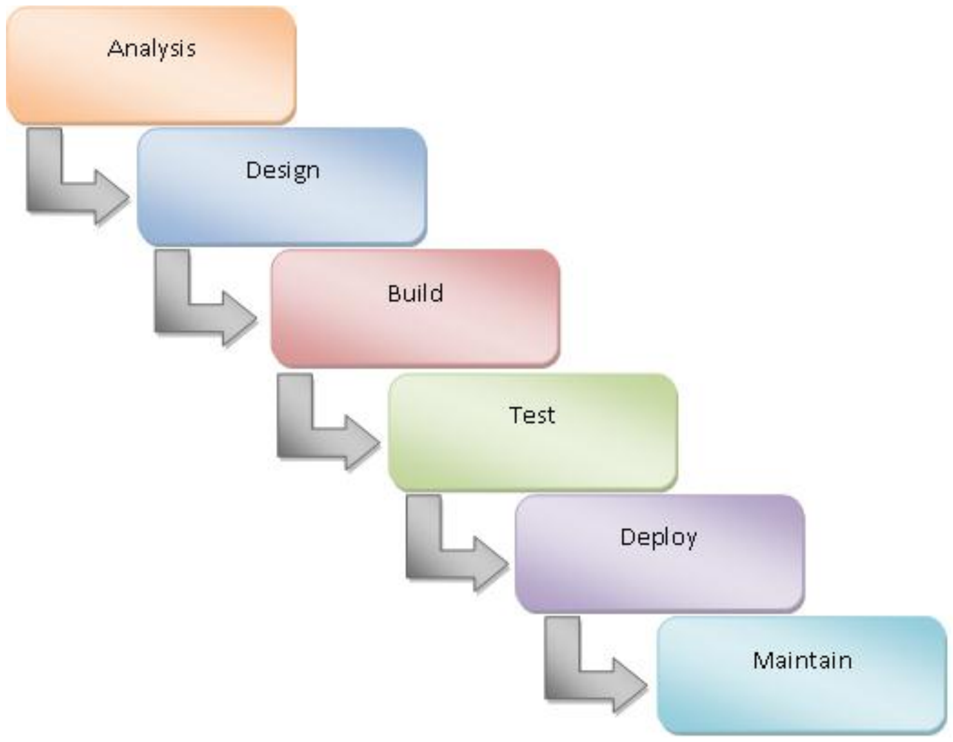


Figure 8 The Waterfall Model

The SDLC often provides a pattern for a software developer to deliver working software because the SDLC is recognisable in most software development approaches (Lewis, 2008, p.77; Oates, 2006, p.39; Hausman and Cook, 2010, p.165; Satzinger et al., 2008, p.40). In Section 2.4.4 the responsibilities of a software developer in relation to each phase of the SLDC are discussed.

2.4.4 Responsibilities of the software developer within the SLDC

A software developer is responsible for designing and implementing an executable code solution, testing the resulting components, and analysing runtime profiles to debug errors that might exist. A software developer may also be responsible for creating the software's architecture and/or employing rapid application development tools (Carroll and Daughtrey, 2007, p.97; Westfall, 2008, p.182).

The following table summarises the key activities and deliverables required from software developers per phase of the SDLC.

Table 4 Activity mapping for software developer on the SDLC adapted from (Wakefield and Thind, Sept. 25-27, 2006)

Phase	***Key Activities***	***Key deliverables***
Analysis	• Understanding the problem • Developing of system objectives and scope • Get high-level user requirements • Conduct system feasibility analysis	• System Features • Feasibility results
Design	• Create process flows and data flows • Develop database design and data models • Architect high-level system function design • Create system function design • Create deployment strategy	• Process models • Data models • Database design • System architecture
Build	• Write program specifications • Code software: • System • Database • Interface • Do unit testing • Do code reviews and walkthroughs	• Program specifications • Source code • Unit testing results
Test	• Track and resolve system defects and issues	• Test issues log
Deploy	• Develop deployment/cutover plan • Develop contingency plan • Perform data conversion • Deploy software/application	• Deployment plan • Production system/code • Contingency Plan
Maintain	• Perform day-to-day support and operations • Correct system defects • Update system documents	• Updated documentation • System defects logs

The activities represent responsibilities that are specific to following the SDLC; however, reality dictates that many software developers are assigned daily tasks and general responsibilities that fall outside of the scope of the SDLC. The next section highlights some of these additional responsibilities in order to demonstrate the wide range of tasks and responsibilities assigned to software developers.

2.4.5 General software developer responsibilities

Very often, software developers are assigned a number of additional duties relating to their general responsibility of designing software systems. What the software developer does on a daily basis will depend on the specific job position with associated responsibilities (Bogue, 2005, p.1; Dooley, 2011, p.1; Kelly, 2008, p.203; Schwalbe, 2010, p.267).

Generally, developers are assigned the responsibility of *developing* a software system from initial design, through graphical user interfaces, creation and coding, to testing and deployment (Bogue, 2004, p.1). It is assumed that a software developer must have the ability to do any kind of development work including:

- database development (Simant, 2009, p.35; Taylor and Parish, 2009, p.189; Khosrow-Pour, 2006, p.3; Hentzen and Nowak, 2002, p.343);
- creating access methods for accessing data in a database (Simant, 2009, p.35; Taylor and Parish, 2009, p.189; Khosrow-Pour, 2006, p.3; Hentzen and Nowak, 2002, p.343);
- troubleshooting performance issues and debugging (Simant, 2009, p.35; Taylor and Parish, 2009, p.189; Khosrow-Pour, 2006, p.3; Hentzen and Nowak, 2002, p.343);
- using and developing technical algorithms (Simant, 2009, p.35; Taylor and Parish, 2009, p.189; Khosrow-Pour, 2006, p.3; Hentzen and Nowak, 2002, p.343);
- using advanced techniques of data processing (Simant, 2009, p.35; Taylor and Parish, 2009, p.189; Khosrow-Pour, 2006, p.3; Hentzen and Nowak, 2002, p.343);
- coding and development of memory management schemes (Simant, 2009, p.35; Taylor and Parish, 2009, p.189; Khosrow-Pour, 2006, p.3; Hentzen and Nowak, 2002, p.343);

- making use of tokenization and compression (Simant, 2009, p.35; Taylor and Parish, 2009, p.189; Khosrow-Pour, 2006, p.3; Hentzen and Nowak, 2002, p.343);
- building encryption and optimization algorithms (Simant, 2009, p.35; Taylor and Parish, 2009, p.189; Khosrow-Pour, 2006, p.3, Hentzen and Nowak, 2002, p.343);
- reading requirements and speaking with customers, internal or external (Simant, 2009, p.35; Taylor and Parish, 2009, p.189; Khosrow-Pour, 2006, p.3; Hentzen and Nowak, 2002, p.343);
- identifying solutions that have not been explored, potential disconnects, and other logical loops (Simant, 2009, p.35; Taylor and Parish, 2009, p.189; Khosrow-Pour, 2006, p.3; Hentzen and Nowak, 2002, p.343);
- ensuring the system interface is intuitive, friendly, and aesthetically appealing (Simant, 2009, p.35; Taylor and Parish, 2009, p.189; Khosrow-Pour, 2006, p.3; Hentzen and Nowak, 2002, p.343);
- facilitating the conversion of data into information (Simant, 2009, p.35; Taylor and Parish, 2009, p.189; Khosrow-Pour, 2006, p.3; Hentzen and Nowak, 2002, p.343);
- using specialized report development tools (Simant, 2009, p.35; Taylor and Parish, 2009, p.189; Khosrow-Pour, 2006, p.3; Hentzen and Nowak, 2002, p.343);
- evaluating the requests of business users (Simant, 2009, p.35; Taylor and Parish, 2009, p.189; Khosrow-Pour, 2006, p.3; Hentzen and Nowak, 2002, p.343);
- extensive testing (Simant, 2009, p.35; Taylor and Parish, 2009, p.189; Khosrow-Pour, 2006, p.3; Hentzen and Nowak, 2002, p.343); as well as
- creating test scripts, or programs, as needed to test the program, to simulate user testing (Simant, 2009, p.35; Taylor and Parish, 2009, p.189; Khosrow-Pour, 2006, p.3; Hentzen and Nowak, 2002, p.343).

It is often stated that *software developers* are the people creating the software that the other roles only influence (Simant, 2009, p.35; Taylor and Parish, 2009, p.189; Khosrow-Pour, 2006, p.3; Hentzen and Nowak, 2002, p.343). From a technical perspective the software developer is at the most basic level expected to be able to translate algorithms and technical specifications into systems that can be executed on a computer system. The language syntax and structures of code to create the system must be understood (Simant, 2009, p.35; Taylor and Parish, 2009, p.189; Khosrow-Pour, 2006, p.3; Hentzen and Nowak, 2002, p.343).

But as could be derived from the list above, knowing the syntax of a programming language is only the basic requirement necessary to be a software developer (Bogue, 2005, p.1, Watt, 2004, p.4). Additional skills required include:

- Developing understanding - any person can follow the instructions laid out for them; however, developers make it a point to understand what they're doing so that they can discover possible issues and opportunities for enhancement (Bogue, 2005, p.1; Perry, 2002, p.290);
- Structures and Algorithms Mastery - In software development there isn't one 'right' way to do things since the same problem can be solved in many ways. However, there are ways that are *more right.* Mastering structures and algorithms means that the problem is solved in the most straightforward and standard manner (Bogue, 2005, p.1; Perry, 2002, p.290);
- Specialization - Developers must specialize in one particular area in order to become a subject matter expert (Bogue, 2005, p.1; Perry, 2002, p.290).

It is expected that software developers have prerequisite syntactical and algorithm skills and those developers who have specialized, must have knowledge of special tools as well (Bogue, 2005, p.1; Perry, 2002, p.290). Software developers are in a constant cycle of building and debugging their code, which relies on problem solving skills. When developing, the software developer does problem solving by figuring out how to get a piece of information that's difficult to get, while during the debugging part of this cycle the developer is focusing on identifying the source of the bug(or bugs), then determining how to eliminate them (Bogue, 2005, p.1; Morley and Parker, 2009,p.549).

In addition, software developers should also be efficient communicators. Many of the individuals who fulfil the role of software developer will liaison with various individuals such as members of the general public and company employees (Bogue, 2005, p.1; Gill, 2011, p.80; Hentzen and Nowak, 2002, p.62). Lastly, software developers should be well versed in various computer programs and methods. A well-rounded software developer is one whose daily job responsibilities will be fulfilled in a fast and efficient manner (Bogue, 2005, p.1; Henderson, 2009, p.465).

Software developers are regarded to be indispensable to the organizations they work for (Bogue, 2005, p.1). They are often the only people who precisely understand how the systems work and reasons for why the system functionality was implemented in a specific way (Bogue, 2005, p.1; Hentzen and Nowak, 2002, p.242). This is specifically true of software developers who are doing maintenance on critical systems, but it can in general be applied to all software developers (Bogue, 2005, p.1; Hentzen and Nowak, 2002, p.242).

Section 2.5 discusses how software developers ensure successful system development as well as what is seen as successful software development. This background is necessary in order to contextualize the responsibilities, which must be considered above and beyond the normal scope of software development as software developers need to keep the success factors in mind when developing software.

2.5 Success factors for software development

In order to answer the question of how many software development projects are completed successfully, the full Software Engineering scope towards system development should be considered. This scope includes budget, schedule, functionality, performance and user satisfaction (Hamilton, 1999, p.311). Budget, schedule, functionality, performance and user satisfaction are factors software developers need to keep in mind to ensure the successful development of systems. The following section discusses these factors.

2.5.1 What are the tasks of a software developer to ensure successful system development?

Software developers need to produce a system that has the desired functionality and performance on time within budget (Westfall, 2008, p.183; Field et al., 1998, p.70).

Software developers should executed the following tasks in order to ensure successful system development:

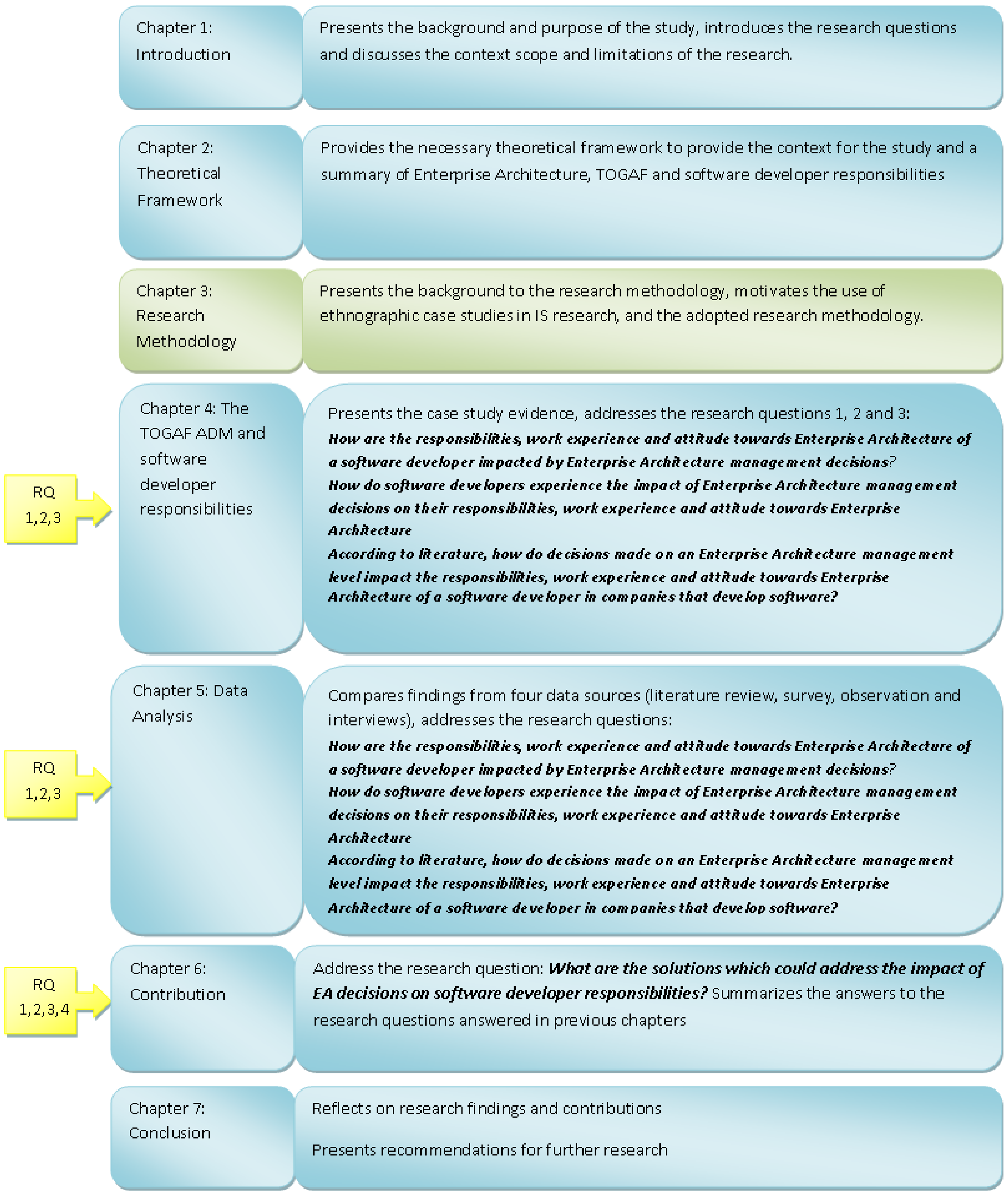
- create and follow a system development plan (McConnell, 2009, p.25; Shelly et al., 2009);
- define the software and requirements baseline, and manage any changes made to this (McConnell, 2009; Shelly et al., 2009, p.25);
- periodically revise system and project health and make adjustments when necessary (McConnell, 2009; Shelly et al., 2009,p.25);
- periodically re-estimate the system size and complexity, the baseline effort estimation and maintenance schedules (McConnell, 2009; Shelly et al., 2009, p.25);
- work through projects and changes systematically (McConnell, 2009; Shelly et al., 2009);
- Set and communicate reasonable goals (McConnell, 2009; Shelly et al., 2009, p.25);
- not relax standards or take short cuts in order to meet a deadline or reduce costs (McConnell, 2009; Shelly et al., 2009, p.25).

When considering the above tasks, it is evident that there are more expectations from software developers when developing a system than just writing the code to implement the system. Some of these factors and soft issues influence the experience software developers have at work. This is important in this study due to the fact that some of these factors are influenced by the decisions made on an Enterprise Architecture management level, and it is necessary to understand why these factors have an influence on the work of a software developer.

2.6 Summary

In this chapter the focus was on Enterprise Architecture, the history of Enterprise Architecture and specifically TOGAF. The TOGAF ADM was discussed in detail, and a mapping of the phases and activities of the ADM provided. Responsibilities of software developers and the SDLC were discussed and the responsibilities of the software developer per SDLC phase provided. Lastly, the activities of software developers to ensure successful software development were discussed. Chapter 3 contains the research methodology, discusses ethnographic research and the adopted research methodology.

Chapter 3: Research Methodology



Chapter 1: Introduction

Presents the background and purpose of the study, introduces the research questions and discusses the context scope and limitations of the research.

Chapter 2: Theoretical Framework

Provides the necessary theoretical framework to provide the context for the study and a summary of Enterprise Architecture, TOGAF and software developer responsibilities

Chapter 3: Research Methodology

Presents the background to the research methodology, motivates the use of ethnographic case studies in IS research, and the adopted research methodology.

RQ 1,2,3

Chapter 4: The TOGAF ADM and software developer responsibilities

Presents the case study evidence, addresses the research questions 1, 2 and 3:

How are the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer impacted by Enterprise Architecture management decisions?

How do software developers experience the impact of Enterprise Architecture management decisions on their responsibilities, work experience and attitude towards Enterprise Architecture

According to literature, how do decisions made on an Enterprise Architecture management level impact the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer in companies that develop software?

RQ 1,2,3

Chapter 5: Data Analysis

Compares findings from four data sources (literature review, survey, observation and interviews), addresses the research questions:

How are the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer impacted by Enterprise Architecture management decisions?

How do software developers experience the impact of Enterprise Architecture management decisions on their responsibilities, work experience and attitude towards Enterprise Architecture

According to literature, how do decisions made on an Enterprise Architecture management level impact the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer in companies that develop software?

RQ 1,2,3,4

Chapter 6: Contribution

Address the research question: ***What are the solutions which could address the impact of EA decisions on software developer responsibilities?*** Summarizes the answers to the research questions answered in previous chapters

Chapter 7: Conclusion

Reflects on research findings and contributions

Presents recommendations for further research

Figure 9 Chapter 3 Dissertation Chapter Map

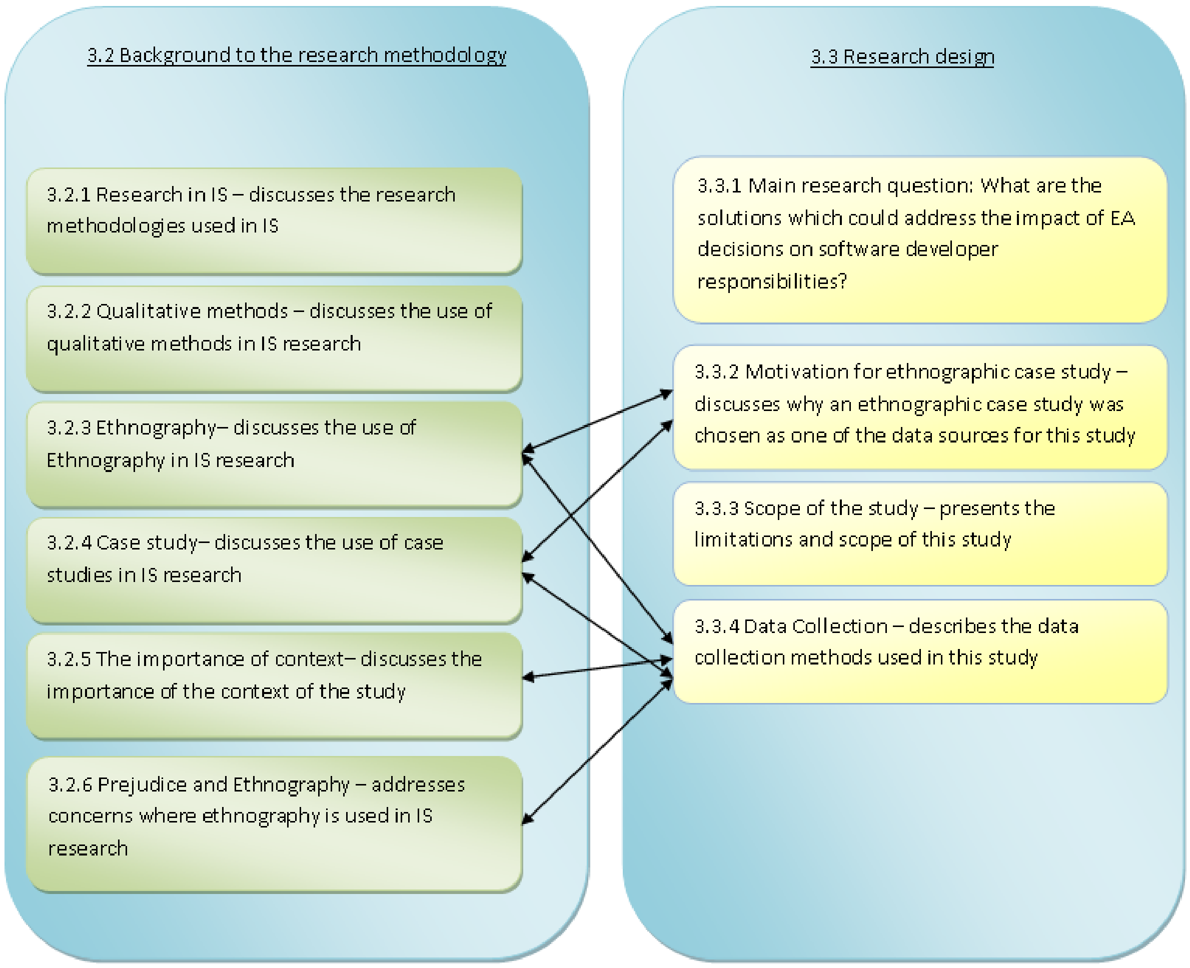

3.2 Background to the research methodology
3.2.1 Research in IS – discusses the research methodologies used in IS
3.2.2 Qualitative methods – discusses the use of qualitative methods in IS research
3.2.3 Ethnography– discusses the use of Ethnography in IS research
3.2.4 Case study– discusses the use of case studies in IS research
3.2.5 The importance of context– discusses the importance of the context of the study
3.2.6 Prejudice and Ethnography – addresses concerns where ethnography is used in IS research
3.3 Research design
3.3.1 Main research question: What are the solutions which could address the impact of EA decisions on software developer responsibilities?
3.3.2 Motivation for ethnographic case study – discusses why an ethnographic case study was chosen as one of the data sources for this study
3.3.3 Scope of the study – presents the limitations and scope of this study
3.3.4 Data Collection – describes the data collection methods used in this study

Figure 10 Chapter 3 Chapter Map

3.1 Introduction

Since the main focus of a research study is to discover answers that will help explain and achieve the aim and objectives of the planned research, a methodological approach is required to facilitate the research process (Kothari, 2008, p.2; Kumar, 2005, p.210).

The purpose of this chapter is to provide background to the philosophical stances of research, to present an overview of qualitative research in Information Systems and to describe the particular research methodology and design used for this specific research study. The research strategy is described in the context of the research participant selection, the case study, data collection, the interview process, interview questions, interview transcription and interview data analysis.

3.2 Background to the research methodology

A research design is a plan, structure and strategy of investigation so conceived to obtain answers to research questions or problems (Kumar, 2005, p. 84).

As can be seen from the quote above by Kumar, the research design is a prerequisite for any research study. However, each study follows a unique design in order to fulfil the purpose of the study. This section covers background and supporting information to the research methods used in this study. The next section contextualizes how research is performed with regards to Information Systems. Section 3.2.2 discusses qualitative methods and explains why qualitative methods were used in this study. Sections 3.2.3 and 3.2.4 provide background on the ethnographic case study and section 3.2.6 discusses prejudice and ethnography.

3.2.1 Research in Information Systems

Information systems must perform faster and more reliably than ever before in order for companies to stay competitive (Camp, 2004, p.21). For information systems to perform faster, more reliably and better, the information systems ought to respond to a rapidly changing environment by being adaptive and learning from past mistakes (Camp, 2004, p.21). Information Systems as a discipline has a strong design aspect, but overall the discipline is an applied *social* science pertaining to the use and impact of technology (Avison and Pries-Heje, 2005, p.191).

Information systems is considered to be a recent discipline, and as no established research methodologies existed, researchers made use of the expertise from a variety of disciplines including computer science, mathematics, linguistics, economics, political science, ethics, sociology and statistics in order to do research in Information Systems (Avison and Pries-Heje, 2005, p.191). Many disciplines, like Information Systems, do not have a simple and single disciplinary status, however, when applied to Information Systems, the research is often criticized for being reactive, opportunistic, lacking academic rigor or being confused (Avison and Pries-Heje, 2005, p.191; King and Lyytinen, 2006, p.9) .

Information Systems, and by inclusion IT concerns rapidly developing applications, and the uses of IT are developing and increasing rapidly (Avison and Pries-Heje, 2005, p.191). Consequently, Information Systems research is characterized by diversity, flexibility and dynamic development (King and Lyytinen, 2006). Because of the dynamic environment of Information Systems, a wide variety of research methods can be used to do research in Information Systems. These methods include qualitative and quantitative methods.

3.2.2 Philosophical perspectives

This section discusses the various philosophical perspectives found in research. Research perspective can be defined as the development of the research background, research knowledge and its nature (Saunders et al., 2009). Research perspective is also defined with the help of a paradigm. "A paradigm is a set of shared assumptions or ways of thinking about some aspects of the world (Oates, 2006, p.282)". The philosophical perspectives included in Information Systems research are positivism, realism and interpretivism.

- *Positivism* means that "the claims for the truth had to be verified empirically. If *something* cannot be verified, its non-sense by definition (Potter, 1996, p.264)".
- *Realism,* which is also referred to as critical realism, "seeks to gain objectivity, however partial or provisional, in and through the subjective elements of human knowing". Researchers using critical realism as a research philosophy believe that the reality we perceive now can be altered later (Bryman and Bell, 2007, p.18).
- *Interpretive* philosophy "emphasizes *understanding* rather than explanation (Carpenter and Arizona State, 2008, p.23)". This philosophy focuses on interpreting the meaning of the study.

The choice of the philosophical perspective and the use of research approach, strategy and data collection method results in findings that match the chosen research perspective (Oates, 2006).

There are research perspectives that better match specific research approaches, research strategies and data collection methods. The choice of the research perspective should be driven by the research being done (Oates, 2006).

3.2.3 Research approach

Research is a systematic way to gain new information (Gliner and Morgan, 2000, p.4). The research approach is the systematic way in which the research is carried out. The research approach contains two approaches: deductive and inductive.

An *inductive* approach deduces scientific laws from particular facts or observational evidence (Mills et al., p.457; Ian, 2009, p.23) Inductive study starts from a specific viewpoint and works toward a more general viewpoint. Inductive research consists of theory from generalized data (Blaikie, 2009, p. 154).

The processes included in an inductive approach are (Trochim, 2006):

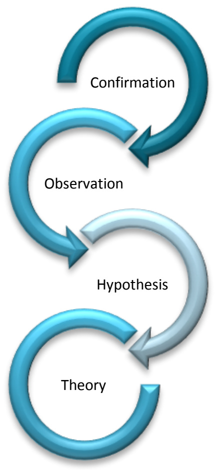


Figure 11 Inductive Study Process

The *deductive* research approach means "to traverse from the large to the small (Lockstrom, 2007, p.79; Ian, 2009, p.24)". "Hypotheses are derived from theory and the tested using empirical methods (Buddenbaum and Novak, 2001, p.12)". Deductive study starts from a general viewpoint and works towards a more specific viewpoint.

The processes included in an deductive study are (Trochim, 2006):

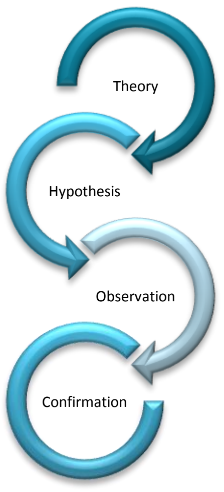


Figure 12 Deductive Study Process

3.2.4 Qualitative and Quantitative Methods

Two types of data analysis are recognised in information systems research, namely quantitative and qualitative analysis (Oates, 2006). Quantitative data analysis makes use of statistical and mathematical formulae and tools to quantify research findings (Oates, 2006). Qualitative data analysis “looks for themes and categories within words people use …” (Oates, 2006, p.38). Qualitative analysis is suited to interpretivistic research as it is the richness and meaning of data that is required (Lee et al., 1997, p.421; Patton, 2002, p.115).

> *The viewpoint holds that information systems practice occurs out there in the real world, whilst researchers observe and reflect from outside (Cater-Steel and Al-Hakim, 2009, p. xviii).*

The above quote by Cater-Steel and Al-Hakim states that the traditional viewpoint of research in *information systems* is to *observe* from outside what is being researched. However, within the information systems discipline, there is a strong background of empirical research to conduct relevant and rigorous studies (Cater-Steel and Al-Hakim, 2009, p. xviii).

Within the context of this study empirical research is “concerned with, or verifiable by *observation* or *experience* rather than theory or pure logic (Dictionary, 2012)”. This is emphasized by the use of participant observation as a data collection method in this study.

Qualitative research includes the use of qualitative data such as interviews, documents and *participant observation* to describe social phenomena (Camp, 2004, p.113). Because the focus in information systems is shifting from the *technological* to *managerial and organizational* issues, qualitative research methods are becoming more important in the field of Information Systems research (Camp, 2004, p.113). Participant observation is often used in Information Systems research in the form of ethnographic case studies. Case studies are considered more than exploratory when used in Information Systems research, because case studies provide researchers with the availability of more detailed information about a specific phenomenon in a rapidly changing environment (Camp, 2004, p.113; Yin, 2009, p.13).

3.2.5 Ethnography

This section introduces background on ethnography in order to provide the context for how ethnography was used in this study. Ethnography is about

> *… telling a credible, rigorous and authentic story, which gives voice to people in their own local context, typically relying on verbatim quotations and thick description (Fetterman, 2010, p.1).*

Ethnography allows the researcher to either overtly or covertly observe the actions of participants in their everyday contexts without participating in the research being done (Hammersley and Atkinson, 2007, p.211). Data collection methods of ethnography mainly include observation and informal conversations (Hammersley and Atkinson, 2007). The data is unstructured, as ethnography does not follow a fixed and detailed research design, and allows for many of interpretations of the meanings, functions and consequences of human actions and institutional practices (Hammersley and Atkinson, 2007, p.3).

In Information System research, ethnography is a qualitative research method that is closely allied to the case study, with the distinguishing factors being the level of involvement of the researcher (Ellis et al., 2009, p.82). The main contribution of ethnography therefore is that is allowed the researcher to participate in the study, giving a rich insight as to the impacts, which are experienced, through personal experience.

3.2.6 Case Study

A case study focuses on one instance of a thing (Oates, 2006, p.144; Smart, 2009, Taylor, 2006; Yin, 2009).

As stated in the quote of Oates et.al, a case study studies one instance of a specific phenomenon. This one instance is studied in depth using a variety of data generation methods (Smart, 2009; Taylor, 2006; Yin, 2009). A case study gives the ethnographic researcher the opportunity to observe what is being studied in detail and to gather a rich insight into that what is being studied and provides information on the relationships and processes of the object (Oates, 2006, p.35). The following quote by Yin emphasises these aspects of case study research:

A case study is an empirical inquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident. (Yin, 2009, p.13).

According to Oates, case studies allow the researcher to focus on depth, rather than breadth. Focusing on depth allows the researcher to focus on small details, which would normally be missed when doing broader scoped research. It also allows the researcher to see how the object behaves in its natural setting, providing an opportunity for holistic study using multiple sources and methods (Oates, 2006).

There are three types of case studies (Oates, 2006, Yin, 2009):

- An exploratory case study is used to describe the questions or hypotheses to be used in a subsequent study.
- A descriptive study leads to a comprehensive study of a particular phenomenon.
- An explanatory study goes further than a descriptive study in trying to explain why events occurred as they did or particular outcomes occurred.

There are also different focuses when selecting a specific case study. Because case studies focus on in-depth investigation, the correct instance of the case study must be chosen. The choice may be based on:

- *Typical instance*: this case study is chosen because the study is typical of many others and can then be used as a representation of its whole class (Oates, 2006; Cohen et al., 2007, p.254).
- *Extreme instance*: this case study is chosen because the study is not typical of others but provides a deviation with the rule (Cohen et al., 2007, p.254; Oates, 2006).
- *Test bed for theory*: the case that is chosen for the study contains elements that make it suitable for testing an existing theory (Cohen et al., 2007, p.254; Oates, 2006).
- *Convenience*: the case that is chosen for the study has people who have agreed to give access and it's convenient considering time and resources (Cohen et al., 2007, p.254; Oates, 2006).
- *Unique opportunity*: the chance to study something that was not planned and may not happen again (Cohen et al., 2007, p.254; Oates, 2006).

As can be seen from the versatility of case study applications above, there are disadvantages in case study research. Case studies are often criticized for generalization (Yin, 2009). Generalization makes it possible through case studies to draw broader conclusions that are relevant beyond the case itself (Yin, 2009). These generalizations are made up from four parts:

- *Concept*: a new idea or notion that emerges from the analysis (Oates, 2006).
- *Theory*: a clear collection of concepts and propositions with a underlying world view (Oates, 2006).
- *Implications*: suggestions about what might happen in other similar instances (Oates, 2006).
- *Rich insight*: is what is gained from a case study that does not fit into the grouping of a concept, theory or an implication (Oates, 2006).

These generalizations allow researchers to fully grasp concepts, theories and implications of what is being study within the case. Because case studies reveal lots of small details that were not necessarily thought of when doing the initial study, case studies provide the researcher with a rich detailed insight into what is being studied (Oates, 2006). This rich insight gives more validity to the research being done, especially in the realm of Information Systems research, which is more focussed on the *managerial and organizational* issues, qualitative research methods, such as the case studies, are becoming more important in the field of Information Systems research (Camp, 2004, p.113).

3.2.7 The Importance of Context

Within Information Systems research, context is important because Information Systems research currently is focusing on *managerial and organizational* issues, rather than the *technological* issues, the social and historical context of Information Systems research takes on specific significance (Camp, 2004, p.113). Engineering based approaches can lead to the over-emphasis of the design and construction of Information Systems, while insufficient attention is given to the social and contextual aspects of Information Systems (Vidgen and Wood, 2002,p.30). To address the shortcoming of Information Systems research neglecting social and contextual aspects, Information Systems researchers recently recognized the value of ethnographic methods for Information Systems research (Lee et al., 1997, p.278; Avison and Pries-Heje, 2005, p.191).

Ethnographic research combined with case studies are qualitative methods, and both allow the researcher to take into account the context of the research done (Hinkel, 2005, p.178; Buchanan and Bryman, 2009, p.xxxv). Yin (2009, p.13) regards the ability of case studies to deal with the context of the case as one of its particular strengths. The boundaries of a case study are hardly ever well defined, but this is not usually a problem, because the context of the case is at least of as much interest as the case itself (Yin, 2009, p.13).

A qualitative approach allows the contextual aspects to be analyzed and interpreted. Interpretive philosophy “emphasizes understanding rather than explanation” (Carpenter and Arizona State, 2008, p.23), whereas positivist studies state that ‘the claims for the truth had to be verified empirically. If something cannot be verified, its non-sense by definition” (Potter, 1996, p.264).

Because of the qualitative approach used in this study, it allows the researcher to analyze and interpret research findings within context of the use of the Information Systems, an because of this context, these findings take on a different meaning and view, which evolves the understanding of Information Systems research (Khosrowpour, 2005, p.2378).

The importance of context in this specific study is stressed by the fact that experience indicates that the impact of Enterprise Architecture management level decisions on software developers' responsibilities relate more to the organizational environment than the technological issues.

This section discussed the importance of context in Information Systems research, because of the way it influences the findings of the study. The next section addresses the concern of researcher prejudice in an ethnographic case study.

3.2.8 Prejudice and Ethnography

> *Interpretivism is the result of a long history of critique of positivism as the sole basis for understanding human activity (Howcroft and Trauth, 2005, p.244).*

The quote from Howcroft and Trauth states that interpretivist research involves the study of social practices in the *context,* in which they occur(Howcroft and Trauth, 2005, p.245). Because of this close involvement with the research subject, ethnographic techniques and participant observation are chosen as qualitative research methods because they produce 'thick descriptions' of organizational contexts and practices, which emphasizes the perceptions and explanations of human actors (Howcroft and Trauth, 2005, p.245).

Prejudice stems from preliminary or previous knowledge, and interpretivists know that this fulfils a valuable role in human understanding, but highlights the need to differentiate between "true prejudice" that leads to understanding and "false prejudice" that leads to misunderstanding. In interpretive field studies, the researcher fulfils a role similar to that of the participants: observing, interpreting and analyzing. As a result of this involvement, prejudice of the researcher is an issue, which is essential to acknowledge (Meyers, 1999, Yin, 2009).

Meyers (1999) propose a set of seven principles for conducting and evaluating these studies:

Table 5 Principles for conducting and evaluating interpretive field studies in Information Systems (Meyers, 1999, p.72)

Principle Number	***Principle for conducting field research***
1	*The fundamental principle of the hermeneutic circle:* This principle is fundamental to all the other principles and suggests that all human understanding is achieved by iterating between considering the interdependent meaning of parts and the whole that they form.
2	*The principle of contextualization:* Requires critical reflection of the social and historical background of the research setting so that the intended audience can see how the current situation under investigation emerged.
3	*The principle of interaction between the researchers and subjects:* Requires critical reflection on how the research materials were socially constructed through the interaction between the researcher and participants.
4	*The principle of abstraction and generalization:* Requires relating the idiographic details revealed by the data interpretation through the application of principles 1 and 2 to theoretical, general concepts that describe the nature of human understanding and social action.
5	*The principle of dialogical reasoning:* Requires sensitivity to possible contradictions between the theoretical preconceptions guiding the research design and actual findings with subsequent cycles of revision.
6	*The principle of multiple interpretations:* Requires sensitivity to possible differences in interpretations among the participants as are typically expressed in multiple narratives or stories of the same sequence of events under study.
7	*The principle of suspicion*: Requires sensitivity to possible “biases” and systematic “distortions” in the narratives collected from the participants.

The first principle suggests that all human understanding is achieved by changing between considering the interdependent meaning of parts and the whole that they form. The second principle requires a critical reflection of the social and historical background of the research setting and the third principle requires that there is reflection on the interaction between researchers and subjects. Principle number four focuses on abstraction and generalization of the data, which was gathered, while the fifth principle requires sensitivity to possible contradictions between the theoretical preconceptions guiding the research design and actual findings. Principle six deals with possible multiple interpretations among participants and the last principle refer to sensitivity to possible prejudice in the narratives collected from the participants.

Section 3.2 provided background and discussions on how research is done in Information Systems, the philosophical perspectives of research and the qualitative methods used in research. It also discussed ethnography and case studies, and addressed the concern of prejudice in ethnographic studies. The next section discusses the research design of the study, and how the methods discussed in section 3.2 were used in the study.

3.3 Research design

This section discusses the research design as it was applied in this study by presenting the research questions and the data collection methods used to collect data in order to answer the research question. It also provides the motivation from the researcher for the type of study that was done. The section below presents the research questions, and explains why the research questions were asked.

3.3.1 Research Questions

Because of the numerous technical and organizational issues that needs to be addressed during the implementation of Enterprise Architecture, there will be challenges (Saha, 2009, p.27; Andersen et al., 2011, p.27). Some of these challenges will naturally differ according to the environment or context. The purpose of this study was to investigate the impact of Enterprise Architecture management level decisions on the responsibilities, work experience and attitude towards Enterprise Architecture of software developers in companies that develop software.

The primary research question addressed by this study was:

How are the responsibilities, work experience and attitude of software developers impacted by the decisions made on an Enterprise Architecture management level in companies that develop software?

During the research design, the study was divided into four sub questions. The sub questions were of an exploratory nature and served to differentiate and guide the study:

1. *According to literature, how do decisions made on an Enterprise Architecture management level impact the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer in companies that develop software?* The purpose of this question was to identify whether published literature and/or solutions to the study being done existed.
2. *How are the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer impacted by Enterprise Architecture management decisions?* The purpose of this question was to see from an observer point of view, what initial impacts could be identified.
3. *How do software developers experience the impact of Enterprise Architecture management decisions on their responsibilities, work experience and attitude towards Enterprise Architecture?* The aim of this question was to obtain feedback from software developers on how they experience the impact of Enterprise Architecture management decisions on their responsibilities.
4. *What are the solutions that could address the impact of Enterprise Architecture management decisions on software developer's responsibilities?* This aim of this research question was to incorporate feedback and possible solutions from the software developers.

These research questions were used and subsequently answered in the following chapters. Research Question 1 is addressed in Chapter 4. Research Questions 2, 3 and 4 are addressed in Chapter 4, 5 and 6. Chapter 4 provides the case study observations and evidence, as well as the participant observations. Chapter 5 analyses the data found from the interviews and surveys and the research findings that are summarized in Chapter 6.

3.3.2 Motivation for ethnographic case study

The research strategy approach chosen for this study can be described as an ethnographic case study. The purpose of this study was to investigate the impact of Enterprise Architecture management level decisions on the responsibilities, work experience and attitude towards Enterprise Architecture of software developers in companies that develop software. A preliminary scan of the literature provided little documentation on these impacts. Because these impacts are *experienced* by software developers, qualitative methods were used in this study. This study has an interpretive perspective because it is based on the interpretation of individuals in their natural setting (Howcroft and Trauth, 2005,p.245).

As the researcher, my long and close involvement with the case study *Company* and its context, as well as the influence of observation on the study, suggest the suitability of ethnographic study methods. Ethnographic research is “about telling a credible, rigorous and authentic story, which gives voice to people in their own local context, typically relying on verbatim quotations and *thick* descriptions” (Fetterman, 2010, p.1). Because this type of research is iterative, the statement of interest evolved into a statement of purpose, which was subjected to regular examination and reconsideration as the study progressed and new facts were revealed.

This study differs from traditional ethnography with regards to the role of the researcher. Although this study corresponds with ethnography with respect to research methods used and the researcher being immersed in the group being studied, an important aspect of ethnography is for researcher to enter the research as a stranger who is suitably prepared to recognize aspects that are extraordinary or unusual. In the literature on ethnography, there is frequent reference to the *researcher going native* (Wiebe et al., 2009, p.425). This implies that the researcher is a stranger to begin with and becomes native for the purpose of the study. In this study the researcher was already a native, and, as such, potentially lost the ability to differentiate between what is ordinary and what is extraordinary. A non-involved outsider can be more scientific and will be more likely to question what others see as familiar (O'Reilly, 2008, p.113).

However, it can also be argued that because the researcher is known to the group being studied, and is considered “one of them”, the researcher may be included in ways only a *native* will understand the meaning of (O'Reilly, 2008, p.113). The labels auto-ethnography and insider ethnography are sometimes used for ethnography that covers the researcher’s own group or has an autobiographical element (Wiebe et al., 2009, p. 596) .

This study stems directly from a desire to achieve an improved understanding of a phenomenon. The study was undertaken from a purely interpretive perspective; however, the findings include both interpretive and critical elements that could lead to changes to the workplace. The benefits of this could include software developers who are more content at work and because of the increased understanding might embrace Enterprise Architecture. The solutions to these impacts, when applied, are out of the scope of this study.

In summary, this study is a qualitative, interpretive ethnographic case study.

3.3.3 Data Collection

Data collection is concerned with offering ways to plan and execute the research (Olsen, 2011, p.3). Because of this focus of gathering data to answer the research questions, this section explains the data collection methods used and how they were structured during this study.

For the purposes of this study, data was collected from four sources, one of which was secondary, and three of which were primary data sources. The primary data sources included participant observation from the case study, semi structured interviews with open ended questions and a survey. The secondary data source was a literature study (Salkind, 2010, p.1330).

3.3.3.1 The literature study

The literature study was conducted to investigate and establish a theoretical framework for the research. The aim of the literature study was to determine the background of Enterprise Architecture, software developers and the SLDC. The literature study included the TOGAF ADM and software developer's responsibilities, of which the combination is discussed in Chapter 4.

"Participant observation is founded on first-hand experience" (Wolcott, 1999, p.46). This implies that the experience of the researcher within context of the study is an important part of the ethnographic research process. The observations made by the researcher are important to this study for the following reasons:

- providing a motivation for the study;
- made the researcher aware that there might be impacts that are not documented or that have solutions;
- motivated the researcher to find possible solutions for impacts the researcher was experiencing; as well as
- guided the research design of the study.

The literature study is discussed in detail in Chapter 2.

3.3.3.2 Participant observation

Observations are data collection methods that observe what participants actually do (Oates, 2006, p.202). This is not just about seeing participants act within a context; it involves a careful assessment of the environment and the behaviour of the participant under observation. Observation could involve the senses of sight, sound, touch, taste and smell, depending on the context. The researcher could act as an invisible observer or active participant in the research process. Because the initial *awareness* of the problem started with participant observation, these observations were verified with semi-structured interviews. These interviews were conducted with three main goals:

- to verify the observations of the researcher, i.e. the impacts experienced by the researcher were experienced or perceived by other software developers in the company;
- to broaden the study, including observations of others that the researcher might not have been aware of; as well as
- to find possible solutions to the impacts identified.

This list of impacts were re-worked and generalized and compiled into a survey, which comprised of yes/no questions, asking participant whether they agreed to a statement. The survey was sent out to participants *outside* the case context to verify the list of impacts and possible solutions. The case context is discussed in more detail in Section 3.3.3.5 and the list of impacts is presented in Chapter 4.

The next subsections introduce the interviews and the interview process followed in this study, to provide context to how the interviews were conducted and how the data obtained from the interviews are used in this study.

3.3.3.3 Interviews

This study used semi-structured interviews utilizing open ended questions. The main purpose of the interviews was to uncover facts and determine participants' perspectives on the impacts of Enterprise Architecture management level decisions on software developer responsibilities, work experience and attitude towards. The responses of the interviews questions contributed to the contextualization of the research context.

Process

Software development has many acronyms and technical jargon, which is commonly understood by individuals in this field. Some of this jargon was included in the interview process, but was not seen as detrimental to the interview process. In order to simplify the use of interviews, the participants were instructed to speak with the terms they use on a daily basis. Participants were encouraged to ask for clarification if they were uncertain of anything. In most cases, all the participants were interviewed on site; interviews lasted about half an hour to an hour. Interviews were conducted in English, as the *Company* is English and most technical terms are English as well. The interviews were recorded using a digital recorder, which conveniently made folders per interview held. Participants were given the options of remaining anonymous or disclosing their identities, however, the *Company* preferred to remain anonymous, and participants were instructed to speak of "the company". In instances where participants forgot, the *Company*'s name was removed.

Questions

The interview questions were categorized according to three themes:

1. Responsibilities of software developers
2. Enterprise Architecture
3. Impact

These themes are summarized in the tables below, as well as the objective of each set of questions:

Table 6 Interview questions – responsibilities of software developers

#	Interview Questions	Theme	Objective
1	Does the development team work within a software development life cycle with clear boundaries, standards or procedures?	Responsibilities of software developers	Determine general software developer responsibilities
2	If the project plan is followed and developers work a standard work day would it be necessary to put in overtime on a daily basis?		
3	When system components are re-used, is it normally done without the requestor knowing about it?		
4	Do you agree or disagree: it is normally quicker and better to re-write system components than trying to re-use existing code		
5	Do software developers just write the code, or make recommendations for better functionality, system flow or other system components like reporting?		

Table 7 Interview Questions – Enterprise Architecture

#	*Interview Questions*	*Theme*	*Objective*
6	Are there formal processes in place to document all changes to the Enterprise Architecture?	Enterprise Architecture	Determine Enterprise Architecture maturity and decision model
7	Are there different business silo's(departments) which have different data silos, copies of data belonging to other silo's or is each silo responsible for the silo’s data?		
8	Is there an architecture team in place who decides what the best architecture is (systems, network, database etc.)?		
9	Is reporting and merging data from multiple systems is a problem?		
10	Do changes made to a system impact more than one business silo (department)?		
11	Do silos (departments) fight over the ownership of data?		
12	Do multiple systems exist in the business?		
13	Are shared processes documented and do they run smoothly?		

Table 8 Interview Questions - Impact

#	Interview Questions	Theme	Objective
14	Do you agree or disagree: Because changes are never properly scoped or the impact realized software developers spend most of their time improvising a way to make the new changes work, often using workaround ways just to get the work done	Impact	Determine the impact on Enterprise Architecture level decisions on Software developers
15	Because software developers have a time constraint, does poorly architected solutions increase the time spent in development?		
16	Agree or disagree: Usually it's a case of develop it as quickly as possible; the stakeholders do not care if the system is coded properly.		
17	Does the system stabilization time increase when systems are coded as quickly as possible without following in-house development standards?		
18	Agree or disagree: Software developers lose interest in projects because of the fact that they just have to get it done, no matter how they do it		
19	When changes are made to the Enterprise Architecture, does it usually involve late hours and many hours of overtime?		
20	When training a new software developer on the Enterprise Architecture and the corresponding system, are system rules and components clear and understandable?		
21	Agree or disagree: Because changes are not thought out properly, but fit in the architecture, development is sometimes unnecessary and functionality is unused.		
22	When doing system maintenance, is it an easy task or does one maintenance job affect multiple areas of the system?		
23	Are all changes to the architecture, and then by association, the system, thought out carefully and thoroughly?		
24	Are the software developers excited about changes to the system when the architecture is changed?		
25	Do changes to the Enterprise Architecture affect software developers? If yes, how?		

Participant Selection

Participants were selected specifically as those who have the job title of *software developer*, ranging from juniors to seniors. Participants were employed by the *Company* mentioned in the case study as part of the development team. In total, 6 people participated in the interviews.

Transcription

Approximately three hours of interview data was recorded. Each interview discussion with a participant was allocated a separate file. The individual files were transferred to a computer hard disk, renamed and systematically organized. As Yin (2003, p. 92) points out, transcription is 'a process that takes enormous time and energy' and this, for practical purposes, the full interviews were not transcribed and the transcription was limited to excerpts of particular interest.

Data Analysis

According to Reis and Judd content analysis is a method used to extract information from a body of material (usually verbal) by methodically and impartially identifying specific characteristics of the material (Reis and Judd, 2000).

Yates describe a form of content analysis where the data is categorized into concepts that are implied by the data known as open coding (Yates, 2003). The purpose of open coding is to *open* up the data with the aim of identifying concepts within the data.

To accomplish this, a table was set up for each interview question in a Microsoft Word document. A table row was allocated to excerpts of each participant's answer. Each answer was analyzed and Word's highlighter tool was used to classify each segment of text considered to represent a specific and relevant concept. Different colours were used to distinguish between concepts, but no particular meanings were attributed to the colours. A column added to the left of the transcript text was used to record an initial list of the concepts recognized. Each recognized concept generally consisted of a short phrase for example Enterprise Architecture decisions or system development time.

Whereas the purpose of open coding is to break open, or fracture, the data, axial coding aims to reassemble the data by clustering the themes identified into categories. To accomplish the axial coding, an Excel spreadsheet was used to gather the initially identified concept phrases in one column and a second column was used to record the theme relating to each concept. This required a number of iterations for refinement and the Excel data sorting function was useful for grouping the concepts according to theme. Each theme was identified typically by a one or two word title or code, such as processes, overtime or issue.

Each concept in the left column of the transcription tables was then translated into its theme code and recoded in a column to the right of the transcription. The transcripts were then studied again in conjunction with the allocated themes, and the list of themes was revised and categorized. An extra column was appended to the extreme right of the transcription tables and used to record the revised themes, which were then used for discussing the data.

3.3.3.4 Survey

In order to validate the results of the participant observation and the interviews a survey was used. The main purpose of the survey was to validate facts and participants' perspectives on the impacts of Enterprise Architecture management level decisions on software developer responsibilities. The responses of the survey questions contributed to the contextualization of the research context.

Process

Software development has many acronyms and technical terms, which are commonly understood by individuals in this field. Some of these terms or 'jargon' were included in the survey process. The inclusion was not seen as detrimental to the survey process as it represented the common language use of developers. The survey comprised of 25 questions, each with a space available for research participants to leave a comment or to clarify an answer. The survey stated the following prior to completion of the survey, assuring respondents of anonymity if they required it:

Dear Survey Participant

Thank you for your willingness to participate in this research project: The Impact of Enterprise Architecture on a Software Developer's responsibilities. Below is more information regarding this project:

The aim of the project is to research the impact of decisions made on an Enterprise Architecture level on the responsibilities of a Software Developer in an organization/business/enterprise. Participation in this survey is voluntary and anonymous. By completing and returning this questionnaire, you agree to participate in this research and to the publication of the results with the understanding that anonymity will be preserved. Although this is an anonymous survey, space is provided at the end of the questionnaire for contact details of participants who would be prepared to make themselves available for short follow-up interviews.

As mentioned, this is an anonymous survey. There will be no attempt at uncovering your identity or the identity of your organization; or examining the responses on an individual basis. The results of the project will be published, but you may be assured that any information obtained in connection with this study that may identify you will remain confidential and will not be disclosed.

The survey was hosted on an internet platform by a company that performs internet surveys. Participants were given the option of leaving their contact details if they were available for follow up interviews.

Questions

Because the main purpose of the survey was to verify the results obtained from the interviews, the set on questions used in the survey is the same as used in the interview, which was discussed in Section 3.3.3.3.

Participant Selection

The survey was sent out to software developers who do not work for the *Company*, with the following message:

Hi

Below is the link to the survey I require to complete my master's degree, please pass on to as many software developers/Programmers/System architects as possible. It is anonymous.

http://NameOf SurveySite.com?s=LIJJNF_e45a67ed

Dear Survey Participant

Thank you for your willingness to participate in this research project: The Impact of Enterprise Architecture on a Software Developer's responsibilities. Below is more information regarding this project:

The aim of the project is to research the impact of decisions made on an Enterprise Architecture level on the responsibilities of a Software Developer in an organisation/business/enterprise.

Participation in this survey is voluntary and anonymous. By completing and returning this questionnaire, you agree to participate in this research and to the publication of the results with the understanding that anonymity will be preserved. Although this is an anonymous survey, space is provided at the end of the questionnaire for contact details of participants who would be prepared to make themselves available for short follow-up interviews.

As mentioned, this is an anonymous survey. There will be no attempt at uncovering your identity or the identity of your organisation; or examining the responses on an individual basis. The results of the project will be published, but you may be assured that any information obtained in connection with this study that may identify you will remain confidential and will not be disclosed.

Regards,

Judith van der Linde

Data Analysis

A table was set up for each survey question in a Microsoft Word document. A table row was allocated to excerpts of each participant's comment if there was one. Each yes/no answer were recorded and a count applied to the predetermined segment considered representing a specific and relevant concept, which was identified in the coding process of the interview data analysis. These concepts and themes were then used to discuss the data.

3.3.3.5 Case Context

For the ethnographic case study used in this research, it is necessary to describe the case study context. The *Company* that forms the context of this study is a department in a company, which provides financial services to a niche market. This *Company* has varying lines of business but runs from a shared cost centre, one of which is Group and IT services. Group and IT services provide all of the services required for the main IT functions, which are provided by the following departments within the Group and IT services department:

- Technical (Support and Installation);
- Networks;
- Group Systems Architecture;
- New technology;
- SharePoint Services;
- Support ;
- Development;
- Business Analysis;
- Business Intelligence;
- Database Administration;
- System Analysis.

The department followed a customized version of the waterfall software development life cycle:

Table 9 Company Customized System Development Life Cycle

Phase	***Department Responsible***	***Deliverable***
Requirements Solicitation	Business Analysis, Group Systems Architecture	Functional Specification
Design	System Analysis	Technical Specification
Build	Development, Database Administration	Working system
Unit Testing	Development, Database Administration	Deployed system to staging area
Maintenance Acceptance Testing (MAT)	Development, Database Administration, Support, System Analysis	Deployed system to test site
User Acceptance Testing (UAT)	Development, Database Administration, Business Analysis, Group System Architecture, Users	User Acceptance testing sign-off
Implementation	Development, Database Administration, Support	System go live
Stabilization	Development, Database Administration, Support	2 week period where the development team assists support with issues arising from the new development
Maintenance	Support	On-going

The customized version of the waterfall software development life cycle used by the *Company* started with a phase called Requirements Solicitation. The teams involved in this phase are the Business Analysts and Group Systems Architecture developers. The purpose of the Requirements Solicitation phase was to identify the requirements, which would be needed for the current software system changes as specified by the *Company*. The outcome of this phase was a Functional Specification document, which stated what needed to be changed where from a functional software system viewpoint.

The second phase in this customized SDLC is a Design phase. The teams involved in this phase are the System Analyst developers. They take the functional specification prepared in the Requirements Solicitation phase and prepare a Technical Specification, which specifies to the software developer precisely where in the system, which changes need to be applied.

Following the Design phase is the Build phase. The teams involved in the build phase are the development and database administration teams. These teams worked together to produce the software systems and applicable changes to the software systems that was specified in the Technical Specification.

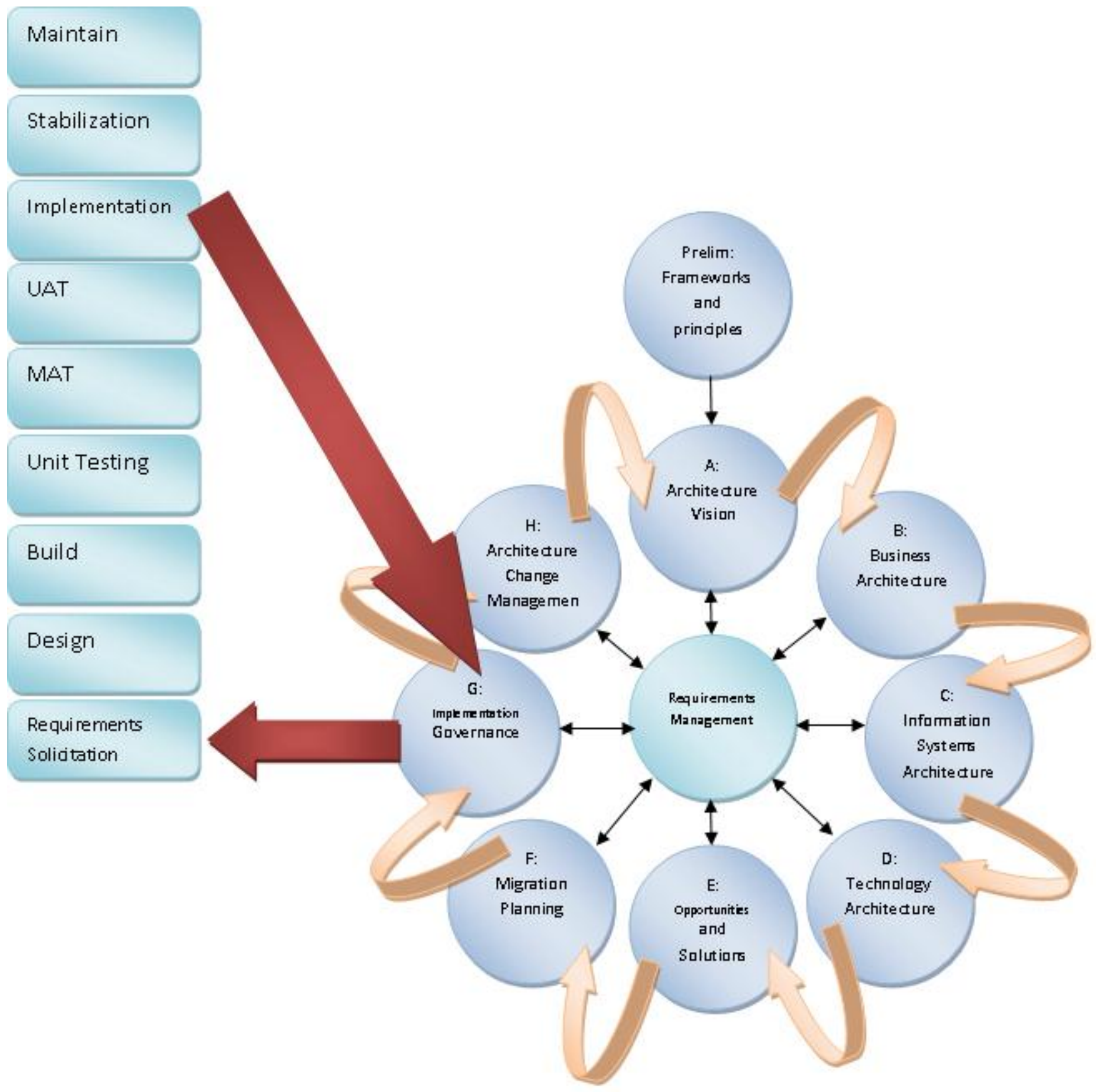
When the Build phase was completed, it was expected that the development and database administration teams perform unit tests on the code to ensure the software development was done according to their understanding of the Technical Specification.

After the Unit Testing phase is completed, the development team delivers the software system to the System Analysis and Support teams, who take the Technical specification and test the software system for the changes made.

When the MAT Phase is completed, the software system is delivered to the Business Analysis and Group Systems Architecture developers who facilitate User Acceptance testing. When the *Company* has signed off on the UAT, the software system is implemented, stabilized for two weeks and then falls into the maintenance category.

This is a settled model and has been in use from before 2006. In 2008, the CEO of the *Company* read an article of Enterprise Architecture, including the range of benefits for the business when Enterprise Architecture is implemented. The *Company* did a study on Enterprise Architecture, and after this investigation agreed with many professionals and scholars that Enterprise Architecture is the solution for most of the problems experienced by organizations and promises to fill the gap where system complexity and poor business alignment is reduced. This is confirmed by John Zachman: *The cost involved and the success of the business depending increasingly on its information systems require a disciplined approach to the management of those systems* (Zachman, 1987, p.276).

After a series of meetings and an investigation into the different Enterprise Architecture frameworks, it was decided that Enterprise Architecture would be implemented and would be the responsibility of the Group Systems Architecture Department. Subsequent to studies and research the Group Systems Architecture team did into Enterprise Architecture Frameworks, the decision was made to slot the current development model into the TOGAF Architecture Development Model. The system development life cycle would then form part of Phase G: Implementation Governance. This forced the *Company* to keep the Enterprise Architecture Repository up to date as all system changes would have to reflect in the repository before the normal system development life cycle would begin. The following figure shows how the SDLC was incorporated into TOGAF.



Maintain
Stabilization
Implementation
UAT
MAT
Unit Testing
Build
Design
Requirements Solidtation
Prelim: Frameworks and principles
A: Architecture Vision
B: Business Architecture
C: Information Systems Architecture
D: Technology Architecture
E: Opportunities and Solutions
F: Migration Planning
G: Implementation Governance
H: Architecture Change Managemen
Requirements Management

Figure 13 The SDLC Incorporated in TOGAF

After the model had been decided and the SDLC incorporated into TOGAF, the teams were ready to start software development on the systems the *Company* used.

The systems that were impacted in this study were customized business solutions, which were built on a generic framework. The software developers working for this *Company* have a certain framework, in which they must operate. This includes the following components:

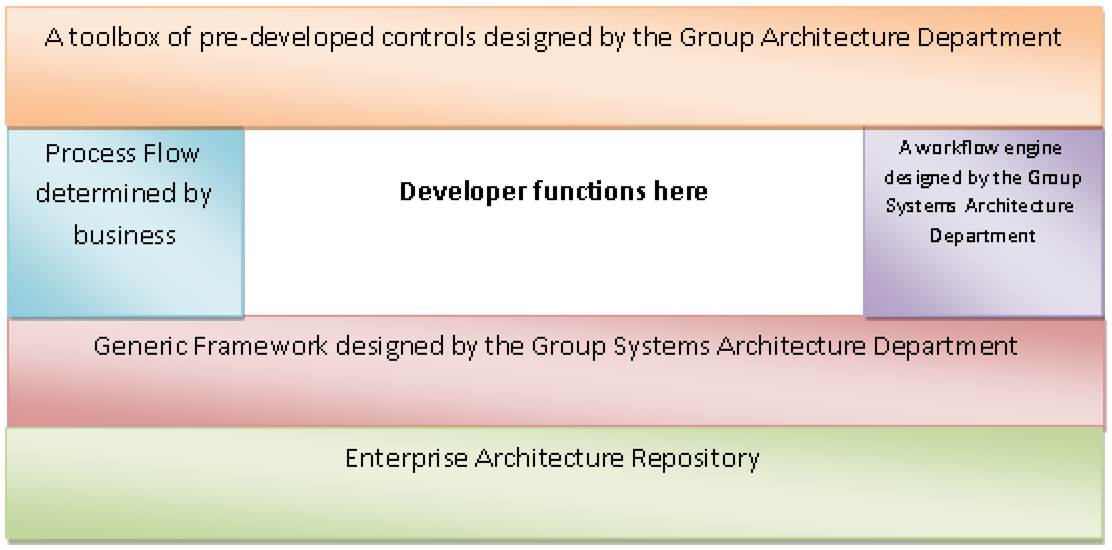

A toolbox of pre-developed controls designed by the Group Architecture Department
Process Flow determined by business
Developer functions here
A workflow engine designed by the Group Systems Architecture Department
Generic Framework designed by the Group Systems Architecture Department
Enterprise Architecture Repository

Figure 14 Software Developer Environment

As is depicted in the figure above, software developers had to work within a set method of operations. This is explained in further detail:

- The Enterprise Architecture Repository is a managed folder solution. Within the repository there resides a multitude of Microsoft Visio files managed by a document management system. The document management system helps with the following in terms of managing the Enterprise Architecture Repository:
    - The integrity of the files – it has change and version control so it can be seen what users changed what items in the repository as well as the date and time the file was changed.
    - Security - only certain people in the architecture team have access to the repository
    - The disaster recovery plan of the repository is implemented easily as all files are stored in one, central place.
    - The files are stored in a central place so everybody knows where to go and get the information.

- The generic framework was developed and maintained by the Group Systems Architecture department. This framework is a custom built in-house framework, which software developers must use in all of their projects. Software developers are not allowed to change this framework as it controls the most important parts of the system architecture. It includes items such as the data access layer and the workflow foundation.

- The generic workflow engine is a process flow engine that runs from the workflow foundation included in the framework and is customized according to the process flow determined by the *Company*.

- The toolbox of pre-developed controls includes controls that are used in more than one place in the system and creates a feel on unity throughout the system. An example of such a control would be a telephone number mask to display numbers in a specific format throughout the system.

The set method of operations provided little room for creativity when doing system scoping and requirements. This method of operations forced a software developer to work in a specific way, which is governed by specific standards and procedures. When changes were made to the Enterprise Architecture, the resulting changes impacted the software developer in such a way that the development team was becoming negative about the work that had to be done. Software developers felt that the Enterprise Architecture would not be 'as bad as it was', but in addition with the set method of operations, it made their work extremely difficult.

This section gave an overview of the *Company* that forms the context of the case study. It also explained the software development process that is followed in the *Company*, and provided some statements of software developers in the *Company*.

3.4 Summary

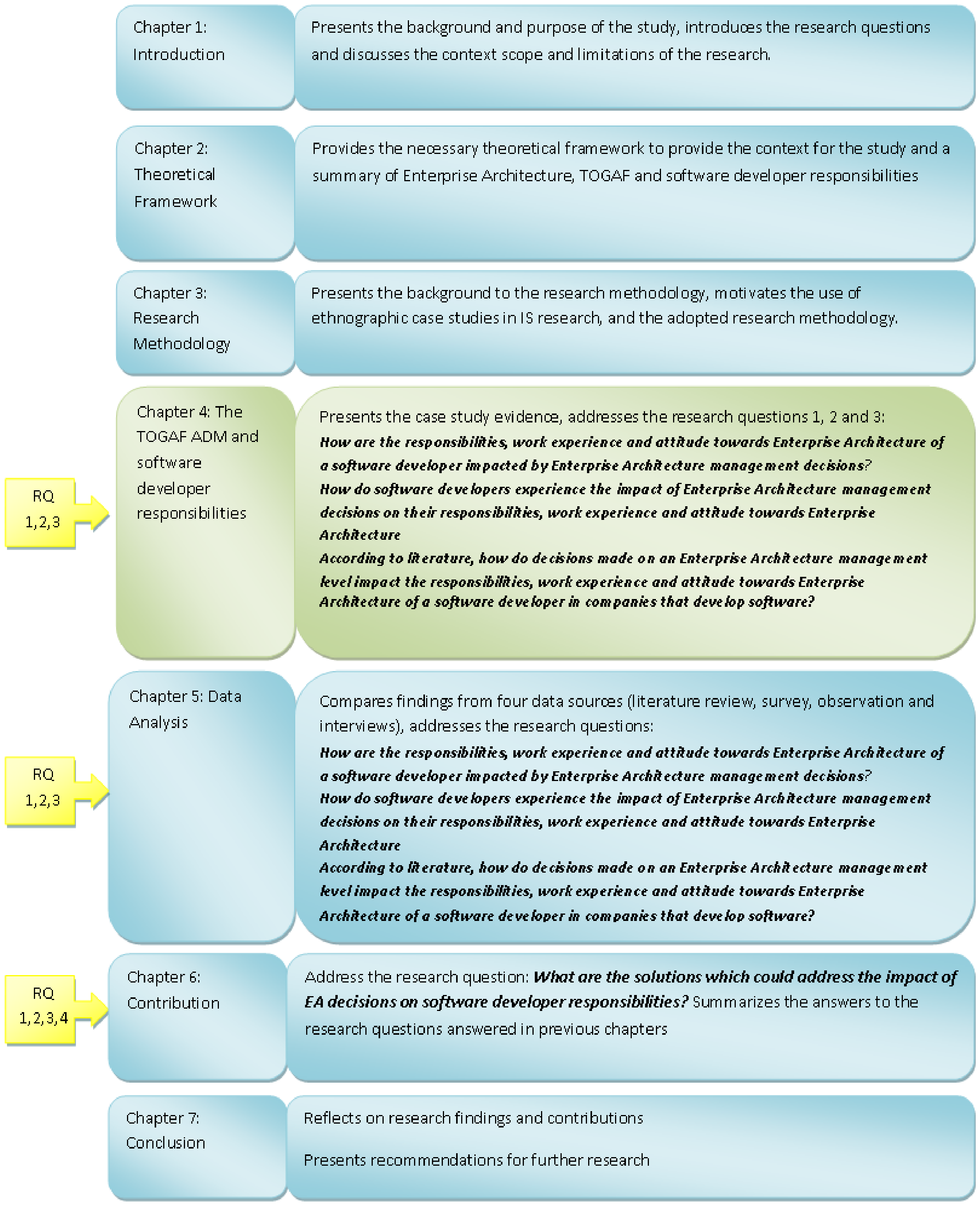
Information Systems is considered to be a recent and dynamic field of study that draws on a wide range of reference disciplines. It is the *soft* issues involving the *managerial and organizational* issues that produce the challenges in Information Systems practice, and thus qualitative methods are complementary to Information Systems research.

Case study and ethnography are examples of qualitative methods commonly used for Information Systems research. These research methods attribute special significance to the research context and the influence of the researcher on the study. This study is an interpretive, case study employing qualitative research methods that are influenced by ethnography.

The context of the study is a small company (referred to as the *Company*) providing financial solutions for a niche market, with a department that focuses on the development of software systems. The purpose was to investigate the impact of Enterprise Architecture management level decisions on the responsibilities, work experience and attitude towards Enterprise Architecture of software developers in companies that develop software. Data was collected by way of a literature study, a set of field interviews, personal experience and a survey. The literature study contributed to the theoretical framework, on which the study was based. The interviews were recorded electronically and key passages were transcribed. The transcriptions were systematically analysed using a basic form of coding.

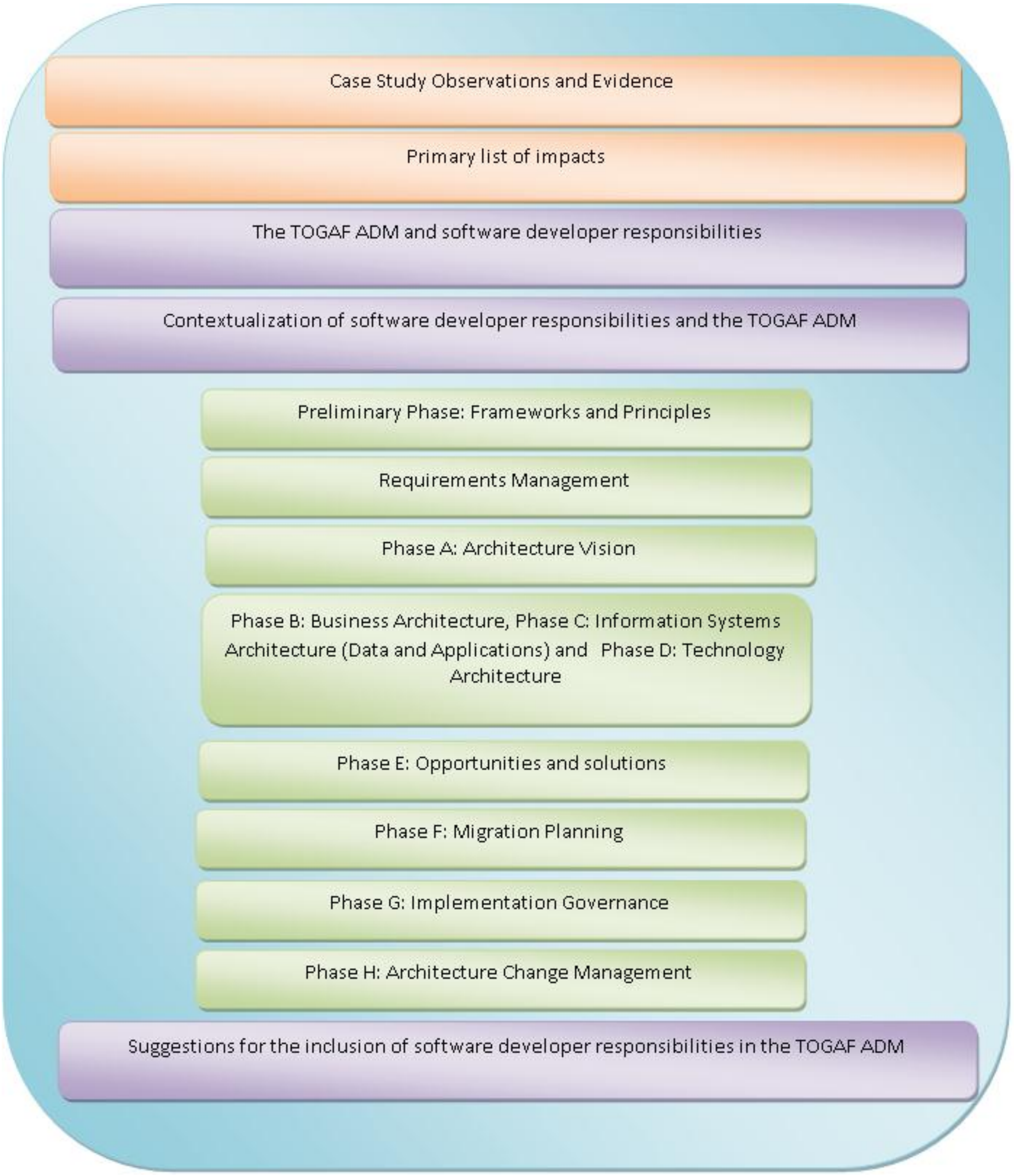
The case study observations are explained in more detail in Chapter 4. The interview findings and participant observations are presented in Chapter 5, while Chapter 6 details the contribution of the study.

Chapter 4: The TOGAF ADM and software developer responsibilities



Chapter 1: Introduction
Presents the background and purpose of the study, introduces the research questions and discusses the context scope and limitations of the research.
Chapter 2: Theoretical Framework
Provides the necessary theoretical framework to provide the context for the study and a summary of Enterprise Architecture, TOGAF and software developer responsibilities
Chapter 3: Research Methodology
Presents the background to the research methodology, motivates the use of ethnographic case studies in IS research, and the adopted research methodology.
RQ 1,2,3
Chapter 4: The TOGAF ADM and software developer responsibilities
Presents the case study evidence, addresses the research questions 1, 2 and 3:
How are the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer impacted by Enterprise Architecture management decisions?
How do software developers experience the impact of Enterprise Architecture management decisions on their responsibilities, work experience and attitude towards Enterprise Architecture
According to literature, how do decisions made on an Enterprise Architecture management level impact the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer in companies that develop software?
RQ 1,2,3
Chapter 5: Data Analysis
Compares findings from four data sources (literature review, survey, observation and interviews), addresses the research questions:
How are the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer impacted by Enterprise Architecture management decisions?
How do software developers experience the impact of Enterprise Architecture management decisions on their responsibilities, work experience and attitude towards Enterprise Architecture
According to literature, how do decisions made on an Enterprise Architecture management level impact the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer in companies that develop software?
RQ 1,2,3,4
Chapter 6: Contribution
Address the research question: What are the solutions which could address the impact of EA decisions on software developer responsibilities? Summarizes the answers to the research questions answered in previous chapters
Chapter 7: Conclusion
Reflects on research findings and contributions
Presents recommendations for further research

Figure 15 Chapter 4 Dissertation Chapter Map


Case Study Observations and Evidence
Primary list of impacts
The TOGAF ADM and software developer responsibilities
Contextualization of software developer responsibilities and the TOGAF ADM
Preliminary Phase: Frameworks and Principles
Requirements Management
Phase A: Architecture Vision
Phase B: Business Architecture, Phase C: Information Systems Architecture (Data and Applications) and Phase D: Technology Architecture
Phase E: Opportunities and solutions
Phase F: Migration Planning
Phase G: Implementation Governance
Phase H: Architecture Change Management
Suggestions for the inclusion of software developer responsibilities in the TOGAF ADM

4.1 Introduction

This chapter discusses the participant researcher's observations with regards to the impact of Enterprise Architecture management decisions on the responsibilities of software developers in the ethnographic case study. In addition, the chapter discusses

Figure 16 Chapter 4 Chapter Map

the mapping of the TOGAF ADM and software developer responsibilities as derived from a literature investigation. This chapter therefore addresses the following three research questions namely:

1. *According to literature, how do decisions made on an Enterprise Architecture management level impact the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer in companies that develop software?* The purpose of this question was to see from an observer point of view, what initial impacts could be identified in the case study.
2. *According to literature, how do decisions made on an Enterprise Architecture level impact the responsibilities of a software developer?* The purpose of this question was to identify whether published literature and/or solutions to the study being done existed.
3. *How do software developers experience the impact of Enterprise Architecture management decisions on their responsibilities, work experience and attitude towards Enterprise Architecture?* The aim of this question was to obtain feedback from software developers on how they experience the impact of Enterprise Architecture management decisions on their responsibilities.

4.2 Case Study Observations and Evidence

A case study focuses on one instance of a thing (Oates, 2006, p.144; Smart, 2009; Taylor, 2006; Yin, 2009).

As stated in the quote of Oates et.al, a case study studies one instance of a specific phenomenon. This one instance is studied in depth using a variety of data generation methods (Smart, 2009; Taylor, 2006; Yin, 2009). A case study gives the ethnographic researcher the opportunity to observe what is being studied in detail and to gather a rich insight into that what is being studied and provides information on the relationships and processes of the object (Oates, 2006, p.35). The following quote by Yin(2009) emphasises these aspects of case study research:

A case study is an empirical inquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident (Yin, 2009, p.13).

According to Oates, cases studies allow the researcher to focus on depth, rather than breadth. This chapter explains the case study evidence and provides a primary list of issues that were identified by means of observation and personal experience.

The initial observation of the problem was gathered from informal discussions with fellow colleagues. During this exercise the researcher gained insight through conversations with other software developers in the case study environment (Spindler and Hammond, 2006, p.34; DeWalt and DeWalt, 2010, p.1; Angrosino, 2008, p.4; Murchison, 2010, p.7). A number of software developers were complaining about issues and the difficulties they have experienced. These complaints increased the researcher's objective to achieve an improved understanding of the impact Enterprise Architecture management decisions have on software developers' responsibilities, work experience and attitude towards Enterprise Architecture of software developers in companies that develop software. There were many discussions with regards to Enterprise Architecture. Software developers were stating the negative impacts of Enterprise Architecture management decisions on software developers' responsibilities, work experience and attitude towards Enterprise Architecture. Examples of these statements include:

I wish the architects would think things through before saying something can be done. I will have to work three hours overtime tonight just to get the component out of the current system it's being used in – Software developer.

This roll out was a big issue. Development had their hands full as the other system components failed when the software was released. They should really learn not to use components from other systems – Support Analyst

I wish my boss would listen to me when I say that it won't work. I told them last time we cannot just copy and paste code because we have that functionality in another system. – Senior Software Developer

The following section presents a primary list of impacts identified through the process of participant observation.

4.3 Primary list of impacts

In observing comments made by colleagues during discussions, the following impacts were captured:

- Software developers are sometimes only notified of a change, not given the opportunity to give advice on what might work better, could be developed faster or a more creative way of doing things
- Not all changes to the Enterprise Architecture are analyzed thoroughly and these changes then create negative occurrences later in the process
- Negative occurrences include the system falling over, data transformation errors, workflow errors, missing data, incorrect system flow, data integrity issues and system concurrency issues.
- All negative occurrences impact the business view of the system, which causes the system to be called unstable, not working, full of errors and bugs and untrustworthy.
- These negative feelings are transferred to the developer, resulting in the software developer being called untrustworthy, writing unstable code and being a ‘bad’ software developer.
- Negative occurrences also take time to fix, and because of the pressure being put on software developers, this results in late hours and working overtime on a daily basis.
- Training new software developers on the system causes difficulties as there is no set structure for changes, thus meaning no solid system architecture exists. Training the support team to maintain the system also results in changes to support procedures, which is normally based on business rules implemented in the system i.e. the business rule changes require that the system must change.

Because of this knowledge gap, software developers are often called in to assist the support team with the challenges they face, which creates more pressure on the software developers who already have time constraints and are putting in overtime. Software developers also feel ‘explaining how the system works’ takes more time than them just fixing the issue themselves.

- The issues experienced when introducing new software developers leads to silos of expertise, as well as cliques of developers. There is the A-team of software developers, the ones who understand everything and can fix everything. A new software developer will take years to reach the system knowledge of a software developer who has been there longer, and with the continuous change, the gap grows daily.
- Having an A-team of developers highlights two areas of risk: software developers who know the system cannot leave without creating a huge knowledge gap and new software developers always feel inadequate; no matter how long they work they will never be the best.
- Software developers also noted that some projects the *Company* required in order to do business, which was placed at the front of the development queue and *had* to be done within half the time it would normally take because it was *business critical,* was never used, or used minimally because the users were not happy with the change.
- Software developers also implemented many of *work-arounds* in order to accommodate the changes to the Enterprise Architecture. These *work-arounds* made maintenance difficult in the sense that one work-around usually broke another work-around.
- These work-around methods are jokingly referred between the software developers as: “Today I *ninja’d* the code again”

There were many other complaints from individuals; however none of them had any relation to Enterprise Architecture and software development, and was thus excluded from the list. This section summarized the primary list of issues, which were identified using participant observation.

These impacts indicate that there are issues with decisions made on an Enterprise Architecture management level. Thus it can be stated that there are misconceptions of how software components are used within the Enterprise Architecture. The most prominent issue is decisions made on the Enterprise Architecture management level has an impact on the software developers especially in the re-use of modules, the workload on software developers, the complexity of systems and the attitude of the software developer. This initial observation of the problem, led the researcher to believe software developers should be included in each phase of the TOGAF ADM during Enterprise Architecture development.

The TOGAF ADM addresses aspects that intuitively influence software development, as was also confirmed by the initial observation from developers in the *Company* listed above. The next section discusses the literature investigation that was done to confirm the observations. The literature investigation aligns the TOGAF ADM and the responsibilities of software developers according to the SDLC.

4.4 The TOGAF ADM and software developer responsibilities

This section provides background from literature on how TOGAF should be linked with the responsibilities of software developers. Literature often states that if the correct SDLC procedures are followed, the development of the required software systems *should* be a success (Horch, 2003, p.240; Burnett, 1998, p.99). It can however be argued that, because Enterprise Architecture causes changes to software systems used in a company, there should be responsibilities for software developers mapped to each phase of the TOGAF ADM, as per the context of this study.

The section therefore addresses the research question:

According to literature, how do decisions made on an Enterprise Architecture level impact the responsibilities of a software developer?

After a preliminary investigation at the commencement of this study it was found that there are many sources on the *impact* of factors on Enterprise Architecture decisions, but limited literature was published on the how decisions made at Enterprise Architecture level impact the individuals and systems within an enterprise. This is regarded as an important deficiency, especially if one considers that Enterprise Architecture promises benefits with regards to the return on investment in IT and the role of individuals in a company to realise the promised return on investment.

The view of the solutions of the alignment and complexity challenges promised by the implementation of an Enterprise Architecture is supported by many studies based on Zachman's paper (Mahmood and Hill, 2011, p.12; Mykityshyn, 2007, p.84; Saha et al., 2009, p.265; Aiguier et al., 2010, p.45). A preliminary search of these studies revealed that most of the published Enterprise Architecture success stories focus on the benefits provided to the company with regards to IT (Beker, 2011, p.245; Dan et al., 2010, p.45; Saha, 2007, p.161; Okunieff et al., 2011, p.17; Ahlemann et al., 2012).

4.4.1 Contextualization of software developer responsibilities and the TOGAF ADM

When considering the specific responsibilities assigned to software developers in the SDLC as discussed in section 2.4.4, it could be argued that it is necessary to assign these responsibilities to the phases of the TOGAF ADM, especially considering the *Company* slotted the SDLC into the TOGAF ADM and due to the deficiency of research on the impact of Enterprise Architecture on the individuals and systems within a company:

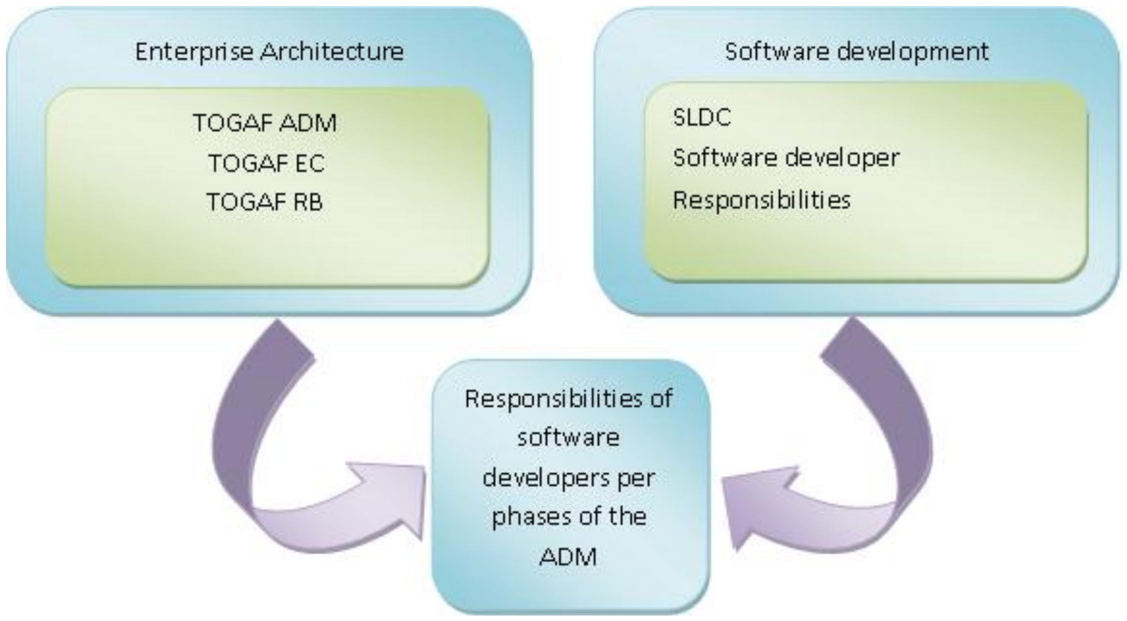


Figure 17 Path to the assigning of software developer responsibilities to the TOGAF ADM

Figure 17 illustrates the argument of the researcher in mapping the TOGAF ADM phases to the responsibilities of software developers. Because each phase of the ADM is concerned with possible system alignment and changes, and these changes are the responsibility of the software developer, and it is plausible to argue that software developers should be involved in the ADM process.

In order to assign software development responsibilities to the ADM, different literature sections should be combined. Figure 17 describes the examination process the researcher followed in recommendation of what software developer responsibilities could be allocated to what phase of the ADM.

Each ADM phase was examined to view the activities which are included in the specific TOGAF ADM phase. Any relevant changes which could affect software developers were correlated to an area in the SDLC which has similar responsibilities. This enabled the researcher to recommend responsibilities for software developers for the phases of the TOGAF ADM. The findings of the examination is summarised in the table below:

Table 10 Suggested software developer responsibilities to be allocated to a TOGAF ADM phase. The ADM phases are adapted from (Blevins et al., 2007, p.25)

ADM Phase	***Activity***	***Suggested SD responsibility***

Preliminary Phase: Frameworks and Principles	The organization must be prepared with the TOGAF procedures in order to facilitate a successful Enterprise Architecture project.	None
Requirements Management	The main expectation of this project is kept here. Each cycle or phase should be validated to the original expectations. Each of the phases store, prioritize, dispose or address requirements, the history is also kept here.	Software developers should work on getting the system in line with the Enterprise Architecture, so expectations of the company should be made available to the software developers so that a bigger picture of the requirement can be formed. This is to facilitate system component re-use to minimize complexity
Phase A: Architecture Vision	The scope, constraints and expectations for the Enterprise Architecture project is setup and defined here. The business context is validated and the Statement of Architecture Work is created.	The full scope and possible future changes should be included for developers to see, this is so developers can plan changes and wider system impacts
Phase B: Business Architecture Phase C: Information Systems Architecture (Data and Applications) Phase D: Technology Architecture	The Enterprise Architecture is developed on the three levels: Business Information Systems Technology This process involves creating a current view of the existing architecture (“as-is”) and the target architecture (“to-be”). This is then followed by a gap analysis to determine what changes should be made to the existing architecture.	When the information systems architecture is done, include the software developer who is going to work on the project specifically for data transfers, code and the “as-is” model. Scoping changes to get to the to-be model should include the developers so they can make suggestions for better solutions
Phase E:	When all the gaps have been	Implement solutions as suggested

Opportunities and Solutions	identified and prioritized, these gaps are sorted into implementation plans.	by the software developers, not as is convenient in the architecture
Phase F: Migration Planning	A cost benefit analysis is done on all changes to be made. An Implementation Road Map is also created noting the analyses and prioritization of the requirements	Software developers should be made aware of the implementation road map, as planning the development and system re-use components are key, this is to ensure that work is not duplicated and functionality unused
Phase G: Implementation Governance	Implementation Architecture Contracts are created to ensure all work is compliant with the current architecture and that work that is carried out conforms to the new architecture.	Unexpected system issues should be addressed here, for instance unforeseen changes, which were not recognized or spotted in the initial design. This allows the software developer to plan changes and deployment carefully to allow for less software issues.
Phase H: Architecture Change Management	Because business's vision changes, the architecture must change to accommodate the change in business. This phase ensures concurrency in the architecture	Software developers need to be aware of all changes made here to be able to evaluate potential impacts in the next architecture iteration

Table 10 as presented above is discussed in more detail in the sections below:

Preliminary Phase: Frameworks and Principles

The organization should be familiarised with the TOGAF procedures in order to facilitate a successful Enterprise Architecture project (Blevins et al., 2007, p.25). Because this phase is a start-up phase and no changes are made to the Enterprise Architecture, there is no need for software developers to be involved in this phase(Raynard, 2008, p.176; Bente et al., 2012, p. 111).

Requirements Management

The main expectation of the project is kept in the requirements management section of the Enterprise Architecture (Bente et al., 2012, p.111; Blevins et al., 2007, p.25; Proper et al., 2009, p.137; Johannesson et al., 2012, p. 135; Raynard, 2008, p.176). Each cycle or phase of the ADM should be validated to the original expectations, which was set at the start of the project. Each of the phases store, prioritize, dispose or address requirements. What happens and changes throughout the iteration of the phases and the relevant changes is also kept here.

This is the central point of all changes to the Enterprise Architecture and software developers should thus be included in this step of the ADM (Bente et al., 2012, p.111; Blevins et al., 2007, p.25; Proper et al., 2009, p.137). Because it will fall within the responsibilities of software developers to work on getting the system in line with the Enterprise Architecture, it will be required of software developers to form a bigger picture of the requirement. This is to facilitate system component re-use to minimize complexity.

Phase A: Architecture Vision

The scope, constraints and expectations for the Enterprise Architecture project is setup and defined here (Bente et al., 2012, p.111; Blevins et al., 2007, p.25; Proper et al., 2009, p.137; Johannesson et al., 2012, p. 135; Raynard, 2008, p.176). The business context is validated and the Statement of Architecture Work is created (Bente et al., 2012, p.111; Blevins et al., 2007, p.25; Proper et al., 2009, p.137; Johannesson et al., 2012, p. 135; Raynard, 2008, p.176).

The full scope and possible future changes should be included for software developers to see in order to facilitate the planning of changes and wider system impacts.

Phase B: Business Architecture, Phase C: Information Systems Architecture (Data and Applications) and Phase D: Technology Architecture

The Enterprise Architecture is developed on the three levels:

- Business;
- Information Systems and
- Technology.

This process involves creating a current view of the existing architecture ("as-is") and the target architecture ("to-be") (Bente et al., 2012, p.111; Blevins et al., 2007, p.25; Proper et al., 2009, p.137; Johannesson et al., 2012, p. 135; Raynard, 2008, p.176). This is then followed by a gap analysis to determine what changes should be made to the existing architecture (Bente et al., 2012, p.111; Blevins et al., 2007, p.25; Proper et al., 2009, p.137; Johannesson et al., 2012, p. 135; Raynard, 2008, p.176).

When the information systems architecture is done, the software developer who is going to work on the project should be included, specifically for data transfers, code and the "as-is" model. Scoping changes to get to the to-be model should include the software developers so they can make suggestions for better solutions.

Phase E: Opportunities and Solutions

When all the gaps have been identified and prioritized, these gaps are sorted into implementation plans (Bente et al., 2012, p.111; Blevins et al., 2007, p.25; Proper et al., 2009, p.137; Johannesson et al., 2012, p. 135; Raynard, 2008, p.176).

The solutions that are recommended to be implemented should take the recommendations of the software developer into consideration, and not just consider what *fits* into the Enterprise Architecture of the company. This is to facilitate better software systems for the company, as the software developers often know the system better than the Enterprise Architect and understand the impact of changes and recommendations.

Phase F: Migration Planning

During this phase a cost benefit analysis is done on all changes to be made. An Implementation Road Map is also created noting the analyses and prioritization of the requirements (Bente et al., 2012, p.111; Blevins et al., 2007, p.25; Proper et al., 2009, p.137; Johannesson et al., 2012, p. 135; Raynard, 2008, p.176).

Software developers should be made aware of the implementation road map. Since planning the development and system re-use components are important, the involvement of software developers is to ensure that work is not duplicated and functionality not used.

Phase G: Implementation Governance

Implementation Architecture Contracts are created to ensure all work is compliant with the current architecture and that work that is carried out conforms to the new architecture (Bente et al., 2012, p.111; Blevins et al., 2007, p.25; Proper et al., 2009, p.137; Johannesson et al., 2012, p. 135; Raynard, 2008, p.176).

Because software systems evolve and there are instances where the Enterprise Architecture is changed in the middle of the project, it could cause unexpected system issues or problems. These unexpected system issues should be addressed during this phase, for instance unforeseen changes that were not recognized or spotted in the initial design. This allows the software developer to plan changes and deployment carefully to allow for less software issues.

Phase H: Architecture Change Management

Because business's vision changes, the architecture must change to accommodate the change in business (Bente et al., 2012, p.111; Blevins et al., 2007, p.25; Proper et al., 2009, p.137; Johannesson et al., 2012, p. 135; Raynard, 2008, p.176). This phase ensures concurrency in the architecture (Bente et al., 2012, p.111; Blevins et al., 2007, p.25; Proper et al., 2009, p.137; Johannesson et al., 2012, p. 135; Raynard, 2008, p.176).

Software developers need to be aware of all changes made during this phase to be able to evaluate potential impacts in the next architecture iteration.

4.4.2 Suggestions for the inclusion of software developer responsibilities in the TOGAF ADM

The research question answered by this section was:

According to literature, how do decisions made on an Enterprise Architecture level impact the responsibilities of a software developer?

The answer to this research question could be created by doing an in-depth study of Enterprise Architecture, specifically TOGAF, and software developer responsibilities, specifically in the SDLC. With a combination of the TOGAF ADM and the SDLC, responsibilities for each phase of the TOGAF ADM could be assigned to software developers.

These responsibilities could include:

- aligning systems with the Enterprise Architecture;
- understand Enterprise Architecture expectations;
- facilitate the development of the Enterprise Architecture;
- advise on possible impacts and changes on software systems when changes are made to the Enterprise Architecture;
- checking the Enterprise Architecture “to-be” model for:
    - data transfer conflicts and issues;
    - code conflicts and issues;
- ownership of the data and how data changes might affect other departments;
- advise of component re-use to minimise system complexity; as well as
- analyse full system impact when changes when changes are made to the Enterprise Architecture.

4.5 Summary

The research questions addressed in this chapter were:

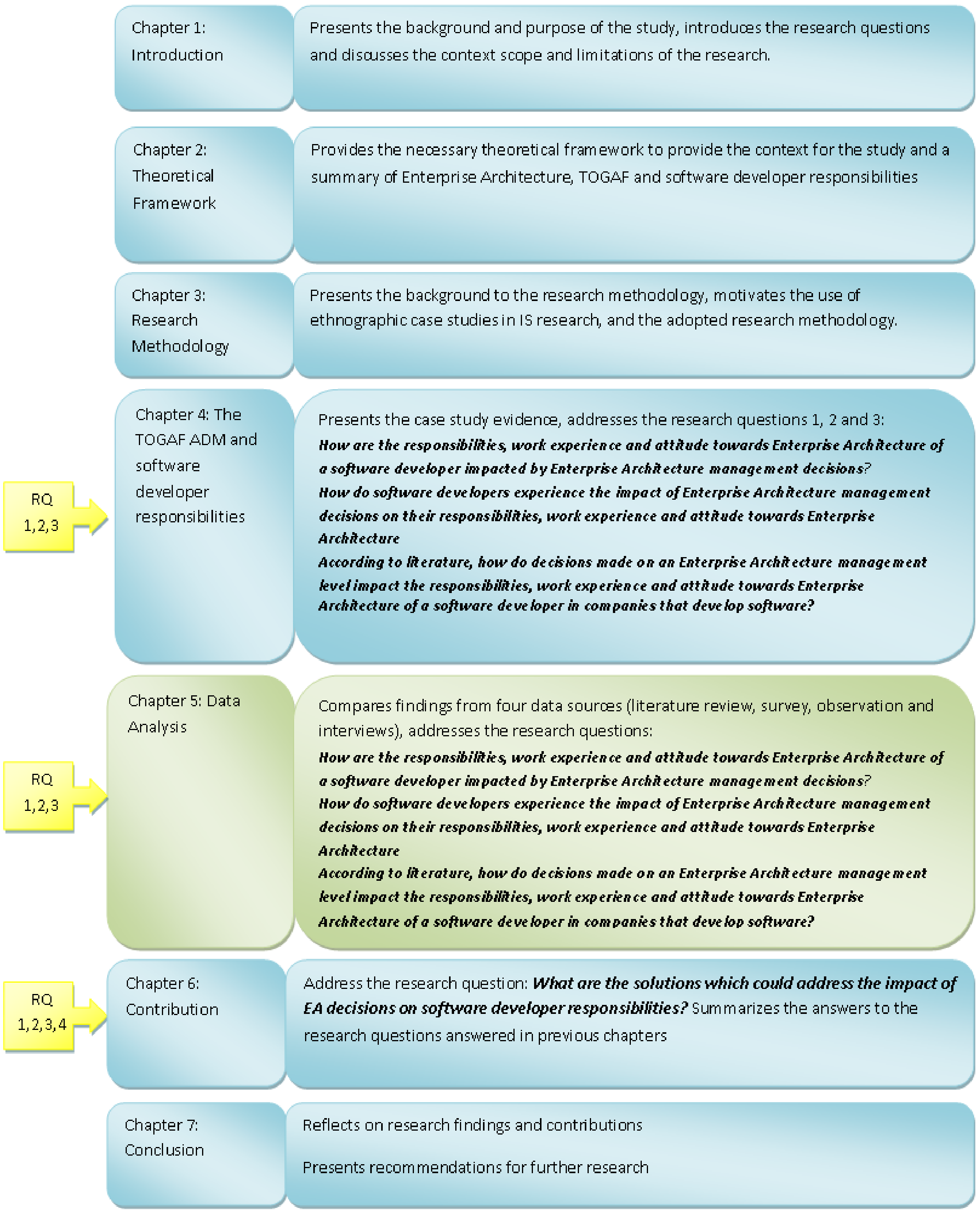
1. *How are the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer impacted by Enterprise Architecture management decisions?*
   In observing comments made by colleagues during discussions, a list of impacts were captured and summarized in Section 4.4.
2. *How do software developers experience the impact of Enterprise Architecture management decisions on their responsibilities, work experience and attitude towards Enterprise Architecture?*

In observing comments made by colleagues during discussions, a list of impacts were captured and summarized in Section 4.3.

3. *How do software developers experience the impact of Enterprise Architecture management decisions on their responsibilities, work experience and attitude towards Enterprise Architecture?* In observing comments made by colleagues during discussions, a list of impacts were captured and summarized in Section 4.3.

This chapter discussed the participant researcher’s observations with regards to the impact of Enterprise Architecture management decisions on the responsibilities of software developers in the ethnographic case study. In addition, the chapter discusses the mapping of the TOGAF ADM and software developer responsibilities as derived from a literature investigation. This chapter also provided suggested responsibilities for software developers mapped to the TOGAF ADM: These responsibilities could include aligning systems with the Enterprise Architecture up to and including analysing full system impact when changes when changes are made to the Enterprise Architecture.

Chapter 5: Data Analysis



Chapter 1: Introduction | Presents the background and purpose of the study, introduces the research questions and discusses the context scope and limitations of the research.

Chapter 2: Theoretical Framework | Provides the necessary theoretical framework to provide the context for the study and a summary of Enterprise Architecture, TOGAF and software developer responsibilities

Chapter 3: Research Methodology | Presents the background to the research methodology, motivates the use of ethnographic case studies in IS research, and the adopted research methodology.

Chapter 4: The TOGAF ADM and software developer responsibilities | Presents the case study evidence, addresses the research questions 1, 2 and 3: ***How are the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer impacted by Enterprise Architecture management decisions? How do software developers experience the impact of Enterprise Architecture management decisions on their responsibilities, work experience and attitude towards Enterprise Architecture According to literature, how do decisions made on an Enterprise Architecture management level impact the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer in companies that develop software?***

Chapter 5: Data Analysis | Compares findings from four data sources (literature review, survey, observation and interviews), addresses the research questions: ***How are the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer impacted by Enterprise Architecture management decisions? How do software developers experience the impact of Enterprise Architecture management decisions on their responsibilities, work experience and attitude towards Enterprise Architecture According to literature, how do decisions made on an Enterprise Architecture management level impact the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer in companies that develop software?***

Chapter 6: Contribution | Address the research question: ***What are the solutions which could address the impact of EA decisions on software developer responsibilities?*** Summarizes the answers to the research questions answered in previous chapters

Chapter 7: Conclusion | Reflects on research findings and contributions

Presents recommendations for further research

Figure 18 Chapter 5 Dissertation Chapter Map

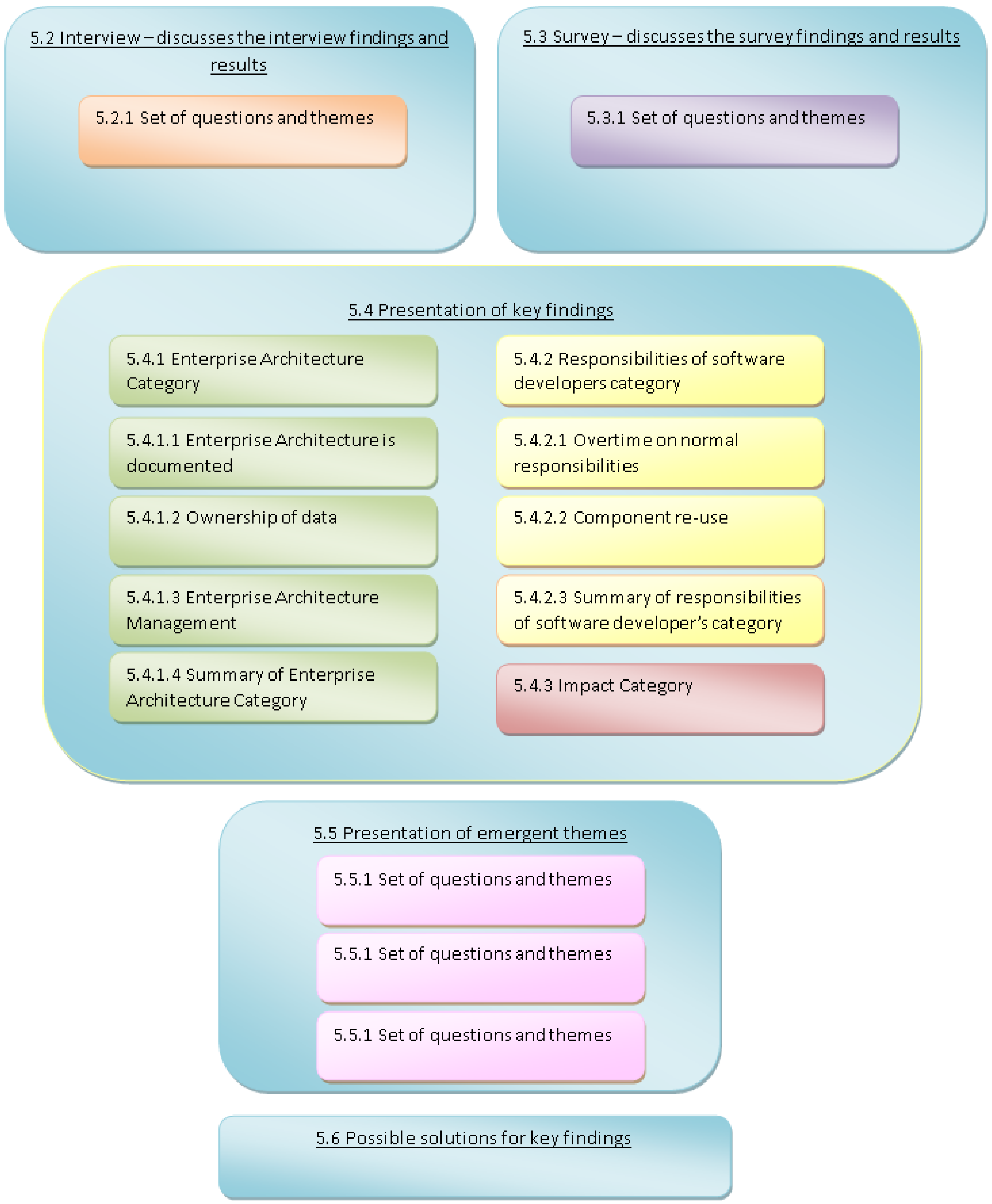

5.2 Interview – discusses the interview findings and results
5.2.1 Set of questions and themes
5.3 Survey – discusses the survey findings and results
5.3.1 Set of questions and themes
5.4 Presentation of key findings
5.4.1 Enterprise Architecture Category
5.4.1.1 Enterprise Architecture is documented
5.4.1.2 Ownership of data
5.4.1.3 Enterprise Architecture Management
5.4.1.4 Summary of Enterprise Architecture Category
5.4.2 Responsibilities of software developers category
5.4.2.1 Overtime on normal responsibilities
5.4.2.2 Component re-use
5.4.2.3 Summary of responsibilities of software developer's category
5.4.3 Impact Category
5.5 Presentation of emergent themes
5.5.1 Set of questions and themes
5.5.1 Set of questions and themes
5.5.1 Set of questions and themes
5.6 Possible solutions for key findings

Figure 19 Chapter 5 Chapter Map

5.1 Introduction

The primary research question addressed by this study is:

How are the responsibilities, work experience and attitude of software developers impacted by the decisions made on an Enterprise Architecture management level in companies that develop software?

The following sub questions are addressed in this chapter and serves to help answer the main research question of this study

1. *According to literature, how do decisions made on an Enterprise Architecture management level impact the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer in companies that develop software?* The purpose of this question was to see from an observer point of view, what initial impacts could be identified.
2. *How do software developers experience the impact of Enterprise Architecture management decisions on their responsibilities, work experience and attitude towards Enterprise Architecture?* The aim of this question was to obtain feedback from software developers on how they experience the impact of Enterprise Architecture management decisions on their responsibilities.

The primary research question is mentioned because it guides the analysis of the data and must be pertinent when looking at the context of the data analysis. Within Information Systems research, context is important because Information Systems research is focusing on *managerial and organizational* issues, rather than the *technological* issues, the social and historical context of Information Systems research takes on specific significance (Camp, 2004, p.113). This significance is important because Information Systems is an interventionist discipline that is concerned with the understanding and formalizing of IT and human interaction with IT (Dwivedi et al., 2011, p. 398). Existing literature shows that it is 'social and organizational contexts of information systems design, development and application, which lead to the greatest practical problems (Dwivedi et al., 2011, p.398).'

This chapter aims to identify the *practical problems*, or as it is contextualized in this study, the *impacts* experienced by software developers from decision made on an Enterprise Architecture management level. This chapter focuses on the data analysis of the impacts of Enterprise Architecture management decisions on software developers, from the transcripts of the field interviews with specific reference to the themes identified during the coding process. The themes are reflected in two main categories, namely the primary themes identified in relation to the interview questions and topic guide, as well as the emergent themes. The emergent themes are themes that evolved during the data-analysis process. Observations and findings in this section are illustrated by using quotes from interview participants where applicable and appropriate.

The following section discusses the interviews, provides the questions asked during the interviews and the interview process. It also discusses the primary and emergent themes from the interview data collection method.

5.2 Interviews

This study used semi structured interviews utilizing open ended questions. The main purpose of the interviews was to uncover facts and determine participants' perspectives on the impacts of Enterprise Architecture management decisions on software developer responsibilities, work experience and attitude towards Enterprise Architecture. The responses of the interviews questions contributed to the contextualization of the research context. Because the interview questions must try to answer the research questions, the research questions were as follows:

The primary research question addressed by this study was:

> *How are the responsibilities, work experience and attitude of software developers impacted by the decisions made on an Enterprise Architecture management level in companies that develop software?*

During the research design, the study was divided into four sub questions. The sub questions were of an exploratory nature and served to differentiate and guide the study:

1. *According to literature, how do decisions made on an Enterprise Architecture management level impact the responsibilities, work experience and attitude*

towards Enterprise Architecture of a software developer in companies that develop software? The purpose of this question was to identify whether published literature and/or solutions to the study being done existed.

2. *How are the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer impacted by Enterprise Architecture management decisions?* The purpose of this question was to see from an observer point of view, what initial impacts could be identified.
3. *How do software developers experience the impact of Enterprise Architecture management decisions on their responsibilities, work experience and attitude towards Enterprise Architecture?* The aim of this question was to obtain feedback from software developers on how they experience the impact of Enterprise Architecture management decisions on their responsibilities.
4. *What are the solutions that could address the impact of Enterprise Architecture management decisions on software developer's responsibilities?* This aim of this research question was to incorporate feedback and possible solutions from the software developers.

Questions for the interviews were designed in order obtain participant feedback to the research questions. Interview questions were designed according to three themes:

- Responsibilities of software developers,
- Enterprise Architecture and
- Impact

These themes are summarized in the tables below, as well as the objective of each set of questions:

Table 11 Interview questions – responsibilities of software developers

#	Interview Questions	Theme	Objective
1	Does the development team work within a software development life cycle with clear boundaries, standards or procedures?	Responsibilities of software developers	Determine general software developer responsibilities
2	If the project plan is followed and developers work a standard work day would it be necessary to put in overtime on a daily basis?		
3	When system components are re-used, is it normally done without the requestor knowing about it?		
4	Do you agree or disagree: it is normally quicker and better to re-write system components than trying to re-use existing code		
5	Do software developers just write the code, or make recommendations for better functionality, system flow or other system components like reporting?		

Table 12 Interview Questions – Enterprise Architecture

#	Interview Questions	Theme	Objective
6	Are there formal processes in place to document all changes to the Enterprise Architecture?	Enterprise Architecture	Determine Enterprise Architecture maturity and decision model
7	Are there different business silo's(departments) which have different data silos, copies of data belonging to other silo's or is each silo responsible for the silo's data?		
8	Is there an architecture team in place who decides what the best architecture is (systems, network, database etc.)?		
9	Is reporting and merging data from multiple systems is a problem?		
10	Do changes made to a system impact more than one business silo (department)?		
11	Do silos (departments) fight over the ownership of data?		
12	Do multiple systems exist in the business?		
13	Are shared processes documented and do they run smoothly?		

Table 13 Interview Questions - Impact

#	Interview Questions	Theme	Objective
14	Do you agree or disagree: Because changes are never properly scoped or the impact realized software developers spend most of their time improvising a way to make the new changes work, often using workaround ways just to get the work done	Impact	Determine the impact on Enterprise Architecture level decisions on Software developers
15	Because software developers have a time constraint, does poorly architected solutions increase the time spent in development?		
16	Agree or disagree: Usually it's a case of develop it as quickly as possible; the stakeholders do not care if the system is coded properly.		
17	Does the system stabilization time increase when systems are coded as quickly as possible without following in-house development standards?		
18	Agree or disagree: Software developers lose interest in projects because of the fact that they just have to get it done, no matter how they do it		
19	When changes are made to the Enterprise Architecture, does it usually involve late hours and many hours of overtime?		
20	When training a new software developer on the Enterprise Architecture and the corresponding system, are system rules and components clear and understandable?		
21	Agree or disagree: Because changes are not thought out properly, but fit in the architecture, development is sometimes unnecessary and functionality is unused.		
22	When doing system maintenance, is it an easy task or does one maintenance job affect multiple areas of the system?		
23	Are all changes to the architecture, and then by association, the system, thought out carefully and thoroughly?		
24	Are the software developers excited about changes to the system when the architecture is changed?		
25	Do changes to the Enterprise Architecture affect software developers? If yes, how?		

The three themes, into which the interview questions were grouped, were responsibilities of software developers, Enterprise Architecture and impacts.

The responsibilities of software developers theme refers to the software developer's view of what their duties *should* encompass. The Enterprise Architecture theme refers to the level of Enterprise Architecture maturity in the *Company* and the research participant's involvement with the Enterprise Architecture process. The impact theme refers to the way in which software developers *experience* the impact of Enterprise Architecture management decisions on their responsibilities.

This section discussed the interview questions, the themes surrounding the interview questions and how the interview questions relate to the research questions.

5.2.1 Set of questions and themes

The responsibilities of software developers as aligned to TOGAF are discussed in Chapter 4 of this study. It was important to establish the research participant's individual views of Enterprise Architecture and software developer responsibilities in their particular context. The questions were designed with the themes and context in mind:

The field interviews were based on a fixed set of questions described in Section 5.2. The questions were open-ended and participants were encouraged to discuss their answers. The discussions were recorded and transcribed. Approximately three hours of interview data was recorded. Each interview discussion with a participant was allocated a separate file. The individual files were transferred to a computer hard disk, renamed and systematically organized. As Yin (2009, p. 109) points out, transcription is "a process that takes enormous time and energy". For practical purposes the full interviews were not transcribed, and the transcription was limited to excerpts of particular interest.

The transcripts were systematically analyzed according to the basic coding strategy. To accomplish this, a table was set up for each interview question in a Microsoft Word document. A table row was allocated to excerpts of each participant's answer. Each answer was analyzed and Word's highlighter tool was used to classify each segment of text considered to represent a specific and relevant concept. Different colours were used to distinguish between concepts, but no particular meanings were attributed to the colours. A column added to the left of the transcript text was used to record an initial list of the concepts recognized. Each recognized concept generally consisted of a short phrase for example Enterprise Architecture decisions or system development time.

Whereas the purpose of open coding is to break open, or fracture, the data, axial coding aims to reassemble the data by clustering the themes identified into categories. To accomplish the axial coding, an Excel spreadsheet was used to gather the initially identified concept phrases in one column and a second column was used to record the theme relating to each concept. This required a number of iterations for refinement and the Excel data sorting function was useful for grouping the concepts according to theme. Each theme was identified typically by a one or two word title or code, such as processes, overtime or issue.

Each concept in the left column of the transcription tables was then translated into its theme code and recoded in a column to the right of the transcription. The transcripts were then studied again in conjunction with the allocated themes, and the list of themes was revised and categorized. An extra column was appended to the extreme right of the transcription tables and used to record the revised themes, which were then used for discussing the data.

This approach produced two classes of themes, firstly, those that relate directly to the topics of the interview questions, and secondly, the additional themes that emerged during the analysis.

Each class comprises three categories. The primary themes class consists of the same three categories as the interview questions. The emergent class is organized into Enterprise Architecture and software development:

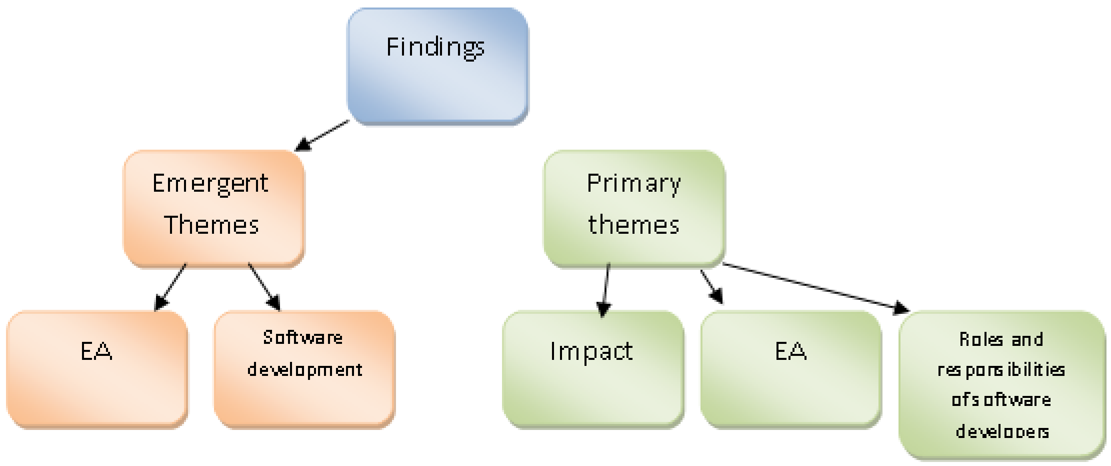


Figure 20 Organization of Data Analysis Theme Categories

5.3 Survey

In order to validate the results of the participant observation and the interviews a survey was used. The main purpose of the survey was to validate facts and participants' perspectives on the impacts of Enterprise Architecture decisions on software developer responsibilities. The responses of the survey questions contributed to the contextualization of the research context.

The research questions and survey questions were discussed in section 5.2. Questions for the survey, were similarly to the interview questions designed in order obtain participant feedback to the research questions. The survey questions were designed according to the three themes that were used for the interview questions namely:

- Responsibilities of software developers;
- Enterprise Architecture and
- Impact.

The definitions of the themes surrounding the survey questions remained as discussed in Section 5.2; the questions used in the survey are the same as were used in the interviews, because the surveys were used to validate the responses received in the interviews.

The themes into which the survey questions were grouped, is the same as the interview questions, namely responsibilities of software developers, Enterprise Architecture and impacts. The responsibilities of software developers theme refers to the software developer’s view of what their duties should encompass. The Enterprise Architecture theme refers to the level of Enterprise Architecture maturity in their company and the research participant’s involvement with the Enterprise Architecture process. The impact theme refers to the way in which software developers experience the impact of Enterprise Architecture decisions on their responsibilities.

This section discussed the survey questions, the themes surrounding the survey questions and how the survey questions relate to the research questions.

5.3.1 Set of questions and themes

The surveys were based on a fixed set of questions described in section 5.2. The questions were yes/no questions and participants were provided for a place to comment on the questions they wanted to. The survey was hosted on an internet survey site and the link provided to participants. This approach produced two classes of themes, firstly, those that relate directly to the topics of the interview questions, and secondly, the additional themes that emerged during the analysis.

Each class comprises three categories. The primary themes class consists of the same three categories as the interview questions.

5.4 Presentation of key findings

In this section, the findings relating to the primary themes are discussed in the following three categories:

- Enterprise Architecture,
- Responsibilities of software developers and
- Impact.

These categories were the according to which the interview and survey questions were grouped. The Enterprise Architecture Category refers to *impacts* identified with regards to the Enterprise Architecture and how this impacts software developers. The

Responsibilities of software developers category refers to *impacts* identified which relate to the responsibilities of software developers, and lastly, the impacts category refers to the impacts which have been identified by software developers. Section 5.4.1 discusses the Enterprise Architecture category.

5.4.1 Enterprise Architecture Category

The literature study (see Chapter 2) established the theoretical framework for the overall research study. To achieve this, it was necessary to investigate the theoretical context of how the TOGAF ADM can be used to develop Enterprise Architecture. Enterprise Architecture is a relative recent phenomenon, which aims to address two prevalent problems in enterprises or companies, namely:

- System complexity—Companies were spending more money building IT systems (Sessions, 2007, p.1 ; Zachman, 1987, p.276; Tupper, 2011, p.26).
- Poor business alignment— Companies were finding it increasingly difficult to keep expensive IT systems aligned with business needs (Sessions, 2007, p.1; Zachman, 1987, p.276; Tupper, 2011, p.26).

To assess the successfulness of Enterprise Architecture, Maturity is used as a metric to identify the success of an Enterprise Architecture Implementation. Ross et al(2006) presented a two-dimensional operating model that depicts levels of Enterprise Architecture maturity. The more mature the company, the better the benefits promised with the use of Enterprise Architecture. This model is comprised of four quadrants that represent different combinations of the levels of business integration and standardization.

The goal is to work from a diversification model to a unification model to align business vision with IT strategy, reducing the overall costs of IT in the business and providing simpler, better and faster solutions to business problems. The Enterprise Architecture category had the following primary themes identified by survey/research question:

Table 14 Enterprise Architecture Category primary themes

Primary Theme	***Question Numbers***	***Section***
Enterprise Architecture is documented	6, 13	5.4.1.1
Ownership of Data	7, 9, 11	5.4.1.2
Enterprise Architecture Management	8, 10, 12	5.4.1.3

The following sections discuss these themes individually.

5.4.1.1 Enterprise Architecture is documented

Enterprise Architecture documentation is one of the measurement factors for Enterprise Architecture maturity (Hanschke, 2010,p. 194). Estimating maturity in Enterprise Architecture management requires an appraisal of content, processes, organization, steering and tool support(Ross et al., 2006, p.47; Hanschke, 2010, p.194). The following aspects are important:

- Completeness – have all models and the interactions between them been documented? Have all parts of the enterprise been documented or just some parts of the enterprise?
- Granularity, up-to-datedness, quality and consistency and
- Ease of maintenance.

To determine the participant’s view on Enterprise Architecture documentation and the completeness of the Enterprise Architecture documentation the following questions were asked:

Are there formal processes in place to document all changes to the Enterprise Architecture?

From the interviews, five out of the six participants stated that there are formal processes in place to document the architecture and to keep the architecture up to date with changes.

From the survey, 55.56% of respondents stated there are no formal processes in place to document the Enterprise Architecture, and two respondents commented on the ease of use and interpretation of the Enterprise Architecture:

- “There are processes in place. However processes are not enforced because people each have their own idea or interpretation of the processes.”
- “There are some documents on the server, but the implementer is finding it difficult to change-manage the people out of their old habits”

The second question pertaining to Enterprise Architecture documentation was:

Are shared processes documented and do they run smoothly?

From the interview 4 of the 6 participants replied negatively. 10 out of 16 survey respondents answered negatively.

This indicates that there is a tendency for companies to have Enterprise Architecture and enforce the Enterprise Architecture, but the Enterprise Architecture is outdated or incomplete. These statements show there is a problem with the Enterprise Architecture documentation, as software developers each interpret the Enterprise Architecture on a different level and there is not a mutual understanding of how the Enterprise Architecture is supposed to be used. It also shows the software developers have habits that are difficult to change. The rationale behind this opinion is another impact identified: Incomplete or partial Enterprise Architecture documentation negatively impacts software developers for the following reasons:

- Incorrect Enterprise Architecture implies incorrect decisions are made on an Enterprise Architecture level that impacts the software developer as well as
- Incorrect decisions on Enterprise Architecture level creates more work for software developers to do, which amounts to overtime being worked, frustration about components not being used and negativity towards Enterprise Architecture.

5.4.1.2 Ownership of data

Enterprise Architecture stipulates an owner assigned to each IT process, which defines the actions and decisions regarding this IT process (Isaca, 2011, p.38). This owner takes sole responsibility for the IT process, even if the process is utilized in many departments.

To determine the participants view, the following questions were asked:

- *Are there different business silo's(departments) which have different data silos, copies of data belonging to other silo's or is each silo responsible for the silo’s data?*
- *Do silos (departments) fight over the ownership of data?*
- *Is reporting and merging data from multiple systems is a problem?*

In answer to the first question the interview participants all concluded that there are different departments that all utilize the shared data. 77.78% of the survey respondents stated there are different silos with data that is shared. One respondent commented:

Yes to different silo’s, No to different data silo’s as all integrated into the same structure, Yes to each department being responsible for their own data.

In answer to the second question three out of six interview participants experienced departments fighting over the ownership of the data, and one software developer commented:

Not that I knew of - but think it did sometimes happen

75% of the survey respondents stated that departments do not fight over the ownership of data.

In answer to the third question, all of the interview participants answered positively stating reporting and merging data from multiple systems is a problem. 56.25% of survey respondents agreed that reporting and merging data from multiple systems is a problem.

This indicates the ownership of data on an Enterprise Architecture level does impact software developers for the following reasons:

- Data ownership poses a problem for software developers when departments have differing views over shared data, because of the resulting changes of department affecting the data and systems of the other department. The software developer is placed in a position between the two departments, each wanting things done their way.
- Software developers spend hours merging and reporting on data from multiple systems in the business, even though Enterprise Architecture dictates the data belongs to one department. This causes overtime and frustration with the software developers.

5.4.1.3 Enterprise Architecture Management

According to the TOGAF framework, Phase H: Because business's vision changes, the architecture must change to accommodate the change in business. This phase ensures concurrency in the architecture (Blevins et al., 2007, p.23).

To determine participant's view on the concurrency of the Enterprise Architecture, participants were asked:

Is there an architecture team in place who decides what the best architecture is (systems, network, database etc.)?

Five of the six interview participants stated there was an Enterprise Architecture team in place who decides on the architecture changes, however, the team has issues implementing the architecture changes and enforcing the Enterprise Architecture. 41.18% of the survey respondents stated there is an Enterprise Architecture team that governs changes to the Enterprise Architecture.

On the question:

Do changes made to a system impact more than one business silo (department)?

Three out of the six interview participants answered yes, while three stated in some instances. 87.5% of survey respondents agreed that changes made to a system impacts more than one department.

The last question in this category:

Do multiple systems exist in the business?

All of the interview participants answered yes and 93.75% of the survey respondents said yes. No participants or respondents commented on this question.

This indicates even though Enterprise Architecture stipulates standardization and unification, multiple systems exist in the business, of which changes impact more than one department and the Enterprise Architecture changes are not concurrent to the actual systems architecture. This impacts software developers for the following reasons:

- In agreement with section 5.4.1.1, non-current Enterprise Architecture is incorrect Enterprise Architecture, thus impacting software developers with incorrect decisions that are made on an Enterprise Architecture level, which creates more work for software developers to do, that amounts to overtime being worked, frustration about components not being used and negativity towards Enterprise Architecture.
- Changes affecting multiple areas to the system, which is not reflected on the Enterprise Architecture, increases development time, makes system maintenance difficult, involves overtime, and relaxes development standards in order for software developers to have “working” systems.

Decisions made on Enterprise Architecture level do not take into consideration these concurrency issues, thus impacting software developers negatively.

5.4.1.4 Summary of Enterprise Architecture Category

Table 15 provides a summary of the findings relating to the Enterprise Architecture category of interview questions. These findings identified three primary themes:

- Enterprise Architecture is documented
- Ownership of data

- Enterprise Architecture management.

Table 15 Summary of Enterprise Architecture category findings

Primary Theme	***Question Numbers***	***Findings***
Enterprise Architecture is documented	6, 13	Incorrect Enterprise Architecture causes incorrect decisions made on an Enterprise Architecture level, which creates more work for software developers to do, which amounts to overtime being worked, frustration about components not being used and negativity towards Enterprise Architecture.
Ownership of Data	7, 9, 11	Data ownership poses a problem for software developers when departments have differing views over shared data, because of the resulting changes of department affecting the data and systems of the other department. The software developer is placed in a position between the two departments, each wanting things done their way. Software developers spend hours merging and reporting on data from multiple systems in the business, even though Enterprise Architecture dictates the data belongs to one department. This causes overtime and frustration with the software developers.
Enterprise Architecture Management	8, 10, 12	Non-current Enterprise Architecture is incorrect Enterprise Architecture, thus impacting software developers with incorrect decisions that are made on an Enterprise Architecture level. This causes changes, affecting multiple areas to the system, which is not reflected on the Enterprise Architecture, increases development time, makes system maintenance difficult, involves overtime, and relaxes development standards in order for software developers to have “working” systems.

Section 5.4.2 discusses the category Responsibilities of software developers' category.

5.4.2 Responsibilities of software developers category

The literature study (see Chapter 2) established the theoretical framework for the overall research study. To achieve this, it was necessary to investigate the theoretical context of which responsibilities are allocated to software developers during the SDLC. Section 5.4.2 discusses the impacts as identified via the coding process described in section 3.3.3. The themes identified are:

- Overtime on normal responsibilities
- Component re-use.

The responsibilities of software developers category is made up of the following primary themes identified per interview/survey question:

Table 16 Responsibilities of software developer's category primary themes

Primary Theme	***Question Numbers***	***Section***
Overtime on normal responsibilities	1,2,5	5.4.2.1
Component re-use	3,4	5.4.2.2

The following sections discuss the themes identified in the Responsibilities of software developer's category:

5.4.2.1 Overtime on normal responsibilities

To determine participants' view on overtime and day-to-day responsibilities of software developers:

> *Does the development team work within a software development life cycle with clear boundaries, standards or procedures?*

All of the interview participants stated that there is a SDLC with clear boundaries, standards and procedures. 50% of survey respondents stated there is an SDLC with clear boundaries, standards and procedures.

The rationale behind the question was to determine whether the SDLC is clearly defined and provides guidance for the normal day-to-day responsibilities of software developers. However, when asked:

If the project plan is followed and developers work a standard work day would it be necessary to put in overtime on a daily basis?

Five of the six interview participants stated that it would not be necessary to work overtime if the project plan – which forms part of the policies and procedures – is followed. One interview participant stated it would not be necessary, but it depends on the software developer's experience. 73.68% of survey respondents agreed that no overtime would be necessary when the project plan is followed.

The last question in this category was:

Do software developers just write the code, or make recommendations for better functionality, system flow or other system components like reporting?

All of the interview participants stated that software developers should be making recommendations for better functionality, as the software developers should know the systems better, but this was not the case as software developers were just given specifications with what to develop. 84.21% of the survey respondents stated that developers should make recommendations.

On the survey the following comments were made:

- "In an environment where the client liaison or the project manager does not know enough about the technology concerned, it is difficult for them to recommend/approve new features or estimate how long it would take to develop them. In these cases it is essential that the programmer provides insight."
- "I feel it is much better for a developer/senior(with programming experience) to make recommendations as they understand the impact of choices"

This indicates that even though the standards and procedures for software development is being followed, the software developers still need to work overtime on specifications provided, in which they have no input. This impacts software developers negatively for the following reason:

- Software developers become work horses who produce code; the impact on other parts of the systems is not realized. This indicates a flaw in the Enterprise Architecture as all components on the Enterprise Architecture level are not clearly defined, as well as the best possible solutions are not being produced because the view of the software developer is not seen as part of the Enterprise Architecture.

5.4.2.2 Component re-use

The core focus of Enterprise Architecture is to move from a diversification model to a unification model, where IT processes are standardized and centralized(Ross et al., 2006), which includes the vision of the company to re-use existing Enterprise Architecture components on a system level. This means software developers need to re-use existing components. To determine participants view on component re-use the following questions were asked:

- *When system components are re-used, is it normally done without the requestor knowing about it?*

- *Do you agree or disagree: it is normally quicker and better to re-write system components than trying to re-use existing code*

On the first question, three of the six interview participants stated system components were being re-used without the requestor knowing about it. Two interview participants said about 50% of the time the requestor knows about it, and one research participant stated that requestors did know about all the component re-use. 63.16% of the survey respondents agreed that system components are being re-used without the requestor knowing about it. A software developer commented:

- “Depends... It does happen where the requestor specifically asks for a component to be re-used... But there are definitely times where the requestor does not know.”

On the second question, three out of the six interview participants stated it is not quicker to re-write system components, where one software developer commented and said: “If the architecture allows it”. 42.1% of the survey respondents agreed that is not quicker or better to re-write system components. 26.32% of the survey respondents were neutral and stated that they neither agree nor disagree with the statement. The following comments were posted on the survey:

- “Customizing an existing system where industries are 80% the same, can work, but out of experience... it is better to re write the system, using new and separate databases than trying to customize an already existing database.”
- “It greatly depends on how unique the components in question are, the more unique they are the more unforeseen obstacles there will be to recreate them. (Especially time spent in discovering the exact features and functionality, to make sure that no benefit previously provided will not be catered for by the replacement.) But assuming that best practices were followed (Utility, Re-usability, Adaptability, Reliability), yes.”

This indicates the software developers do not agree with re-using components, especially if the best practices and in-house development standards are not followed. The negatively impacts software developer because they are forced to re-use components, which means they will spend more time customizing the existing components, which is not guaranteed to fit and work the way the Enterprise Architecture envisions it, resulting in *work-arounds* and a lessening in the quality of the product provided.

5.4.2.3 Summary of responsibilities of software developers category

Table 17 provides a summary of the finding relating to the responsibilities of software developers category of interview questions. These findings identified two primary themes:

- Overtime on normal responsibilities
- Component re-use.

Table 17 Summary of responsibilities of software developers' category

Primary Theme	***Question Numbers***	***Findings***
Overtime on normal responsibilities	1,2,5	Software developers become work horses who produce code; the impact on other parts of the systems is not realized. This indicates a flaw in the Enterprise Architecture as all components on the Enterprise Architecture level are not clearly defined, as well as the best possible solutions are not being produced because the view of the software developer is not seen as part of the Enterprise Architecture.
Component re-use	3,4	Software developers do not agree with re-using components, especially if the best practices and in-house development standards are not followed. The negatively impacts software developer because they are forced to re-use components, which means they will spend more time customizing the existing components, which is not guaranteed to fit and work the way the Enterprise Architecture envisions it, resulting in work-arounds and a lessening in the quality of the product provided.

Section 5.4.3 discusses the impact category. The primary list of impacts was compiled into the interview/survey questions, and as discussed in Section 3.3.3, participants were asked whether they agreed with a certain impact or not. The Impact category discusses these impacts and whether the participants and respondents agreed with the impact or not.

5.4.3 Impact Category

The impact category is made up of the following primary themes which were identified using the coding method described in Section 3.3.3:

- Impact: Scope Creep

- Impact: Improvisation
- Impact: Lessening of development standards
- Impact: Overtime
- Impact: Design flaws
- Impact: Quality of final product
- Impact: Teaching new software developers is difficult
- Impact: Boredom
- Impact: Unused components
- Impact: Increased maintenance

Because multiple *impacts* have been identified per interview question (IQ) asked, the following table summarizes the interview/survey questions with the impacts involved:

Table 18 Impacts per interview/survey question

Impact/IQ	***14***	***15***	***16***	***17***	***18***	***19***	***20***	***21***	***22***	***23***	***24***
Impact: Scope Creep	x										
Impact: Improvisation	x		x						x		
Impact: Lessening of development standards	x		x		x	x					
Impact: Overtime		x	x	x		x	x		x		
Impact: Design flaws		x		x		x	x		x		
Impact: Quality of final product		x	x								
Impact: Teaching new software developers is difficult				x			x	x			
Impact: Boredom					x		x	x			x
Impact: Unused components								x		x	
Impact: Increased maintenance								x	x	x	

The table will be discussed per interview/survey question (IQ #). The first interview question, IQ 14, which pertains to impacts, is discussed below:

IQ 14: Do you agree or disagree: Because changes are never properly scoped or the impact realized software developers spend most of their time improvising a way to make the new changes work, often using workaround ways just to get the work done.

All of the interview participants agreed because changes are never properly scoped or the impact realized software developers spend most of their time improvising a way to make the new changes work, often using workaround ways to get the work done. One interview participant commented:

“I agree – the systems are full of *work-arounds*.”

75% of the survey respondents agreed. Comments made on the survey included:

- “Changes that have an unintended impact are communicated and new scope is formed.”
- “Need to train people to prepare well scoped documents - communication with software developers will also improve the process.”

The impacts identified for the above question are:

- Scope Creep;
- Improvisation; as well as
- Lessening of development standards.

Scope creep was identified as an impact for this question because the participants agreed that changes are never properly scoped. Items that were not properly scoped will have to be reworked or the scope of the original work broadened to accommodate anything that was missed.

Improvisation was identified as an impact because the participants agreed that software developers spent most of their time trying to make the changes work with what is currently in the system and the corresponding Enterprise Architecture.

Lessening of development standards was identified as an impact because participants agreed that software developers use work-around ways to get changes done, and the work is then not completed to best practices and standards.

IQ 15: Because software developers have a time constraint, does poorly architected solutions increase the time spent in development?

All of the interview participants agreed because software developers have a time constraint, poorly architected solutions increase the time spent in development. Two interview participants commented:

- “Sometimes things get broken without it being meant.”
- “Yes - it always breaks when you add a new change.”

100% of the survey respondents agreed. Comments made on the survey included:

- “Poorly architected solutions means taking more time for changes at later stages.”
- “Having to re-work issues is a problem and is time consuming - need to ensure that sufficient resources are available”

The impacts identified for the above question are:

- Overtime;
- Design flaws; as well as
- Quality of final product.

Overtime was identified as an impact for this question because participants agreed that even though software developers work within a time-constraint, poorly architected systems increase the development time needed and this means software developers need to put in extra time to get the system changes done on time.

Design flaws was identified as an impact on this question because participants agreed that poorly architected systems will cause much re-work in the system, especially if changes to the system affect more than one part of the system as discussed in Section 5.4.1.3.

Final quality of product was identified as an impact, because design flaws will lead to a poor quality system when the system goes live. The ‘expertise’ of the software developer is determined by the quality of deliverable of the software developer, and if the quality drops, software developers can be labelled as ‘bad’ software developers.

<u>IQ 16</u>: Agree or disagree: Usually it's a case of develop it as quickly as possible; the stakeholders do not care if the system is coded properly.

All of the interview participants agreed that it's a case of develop it as quickly as possible; the stakeholders do not care if the system is coded properly. 56.25% of survey respondents agreed. Comments made on the survey included:

- "Yes and then we have problems with "*bugs*" and errors later! Clients should be informed of the risk of a "*quick fix*""
- "I agree, but since it's good to have programmers advise on how long something will take, and they know that a poorly coded system will just cost the client more in the future to maintain, it is up to them to enforce the allocation of a proper amount of time."

The impacts identified for the above question are:

- Improvisation;
- Lessening of development standards;
- Overtime; as well as
- Quality of final product.

Improvisation was identified as an impact because the participants agreed that system changes must be provided as quickly as possible, and does not always conform to the best practices, which is why lessening of development standards is also an impact identified by this question. This has a snowball effect leading to a drop in the quality of the final product.

Overtime was identified as an impact because interview participants agreed that all items must be developed quickly, and within the case context that stipulates overtime.

<u>IQ 17:</u> Does the system stabilization time increase when systems are coded as quickly as possible without following in-house development standards?

Four out of the six interview participants stated that the system stabilization time increases when systems are coded as quickly as possible without following in-house development standards. 56.25% of the survey respondents agreed that system stabilization time increases when systems are coded as quickly as possible without following in-house development standards. The impacts identified for the above question are:

- Overtime;
- Design flaws; as well as
- Teaching new software developers is difficult.

Overtime was identified as an impact because software developers already have limited time to do allocated projects. Spending time to teach new software developers how the changes were made and the impacts on the rest of the system takes up development time. Design flaws were identified as an impact because it relates to software being developed as quickly as possible.

IQ18: Agree or disagree: Software developers lose interest in projects because of the fact that they just have to get it done, no matter how they do it.

All of the interview participants agreed that software developers lose interest in projects because of the fact that they *just* have to get it done, no matter how they do it. 56.25% of the survey respondents agreed that software developers lose interest in projects because of the fact that they *just* have to get it done, no matter how they do it. The impacts identified for the above question are:

- Boredom; as well as
- Lessening of development standards.

Boredom was identified as an impact because participants stated that software developers lose interest in projects. Software developers, who continuously get the same tasks because they have the experience to do it quickly, get bored. Lessening of development standards was identified as an impact because if software developers just code to get a system change done quickly, adherence to the development standards will be lessened as best practices take time to adhere to and implement.

IQ 19: When changes are made to the Enterprise Architecture, does it usually involve late hours and many hours of overtime?

All of the interview participants stated that when changes are made to the Enterprise Architecture, it does involve late hours and many hours of overtime. Comments made by interview participants were:

- “A lot of late hours - cause it’s a live system.”
- “Yes with no pay”

56.25% of the survey respondents agreed that when changes are made to the Enterprise Architecture, it does involve late hours and many hours of overtime. The impact identified for the above questions is:

- Overtime.

Overtime was identified as an impact because interview participants agreed that changes to the Enterprise Architecture involves late hours and many hours of overtime.

IQ 20: When training a new software developer on the Enterprise Architecture and the corresponding system, are system rules and components clear and understandable?

Five out of the six interview participants stated that it is difficult to train new software developers on the system as the rules and components within the system are all interrelated and dependant. 56.25% of the survey respondents agreed that it is difficult to train new software developers on the system as the rules and components within the system are all interrelated and dependant. The list of impacts identified for the above questions are:

- Teaching new software developers is difficult;
- Boredom;
- Design flaws; as well as
- Overtime.

Teaching new software developers was identified as an impact because the research participants stated that it is difficult to train new software developers on the system.

Boredom has been identified as an impact because software developers who need to train the new software developers need to spend hours explaining system rules and components that they already know. Boredom has also been identified because of the time it takes new software developers to understand the system components and rules; tasks are allocated to software developers who already know how to do the task because they can accomplish the task quicker. Design flaws have been identified as an impact because when new software developers do not understand the Enterprise Architecture and the corresponding system, they introduce flaws into the system. Overtime has been identified as an impact because software developers still need to finish tasks allocated to them even though they are training new software developers, so they need to get time to get their own work done. Also, when a design flaw has been introduced into the system, more established software developers are called in to fix the flaws on the production system after hours.

IQ 21: Agree or disagree: Because a change is not thought out properly, but fits in the Enterprise Architecture, development is sometimes unnecessary and functionality is unused.

Five out of the six interview participants agreed. Comments made by the interview participants included:

- “Agree - I have spent hours developing something that was later not used.”
- “Agree - especially the reports.”

65% of the survey respondents agreed because a change is not thought out properly, but fits in the Enterprise Architecture, development is sometimes unnecessary and functionality is unused. The list of impacts identified for the above questions are:

- Teaching new software developers is difficult;
- Boredom;
- Unused components; as well as
- Increased maintenance.

Unused components have been identified because interview participants stated some developed functionality is unused. Increased maintenance is a by-product of unused components, the more components there are in the system the more maintenance is required. Boredom has been identified as a risk because some software developers experience their development projects as not being worth the time and effort because of the likelihood that the functionality will not be used. *Teaching new software developers is difficult* has been identified as an impact because of the increasing amount of system components, which need to be taught to new software developers.

IQ 22: When doing system maintenance is it an easy task or does one maintenance job affects multiple areas of the system?

All of the interview participants stated that doing system maintenance affects multiple areas of the system. 75% of the survey respondents agreed that doing system maintenance affects multiple areas of the system. The impacts identified for the above research question are:

- Improvisation;
- Overtime;
- Design flaws; as well as
- Increased maintenance.

Improvisation, overtime, design flaws and increased maintenance were identified as impacts because when maintenance affects multiple areas of the system, a change can inadvertently cause other components to break, which needs to be fixed quickly. The quick fix can then inadvertently break another system component.

IQ 23: Are all changes to the architecture, and then by association, the system, thought out carefully and thoroughly?

Four out of the six interview participants stated that changes to the Enterprise Architecture that translates into system changes are not always thought out thoroughly. 37.5% of the survey respondents agreed that changes to the Enterprise Architecture that translates into system changes are not always thought out thoroughly, while 25% had no opinion. A comment made on the survey was:

- “Yes if properly documented and planned.”

The impacts identified for the above research question are:

- Unused components; as well as
- Increased maintenance.

Unused components were identified as an impact because if the changes are not thought out thoroughly and carefully, the functionality that was provided will not be used if it is not what was required. Increased maintenance was identified as an impact because if the changes to the Enterprise Architecture and corresponding system were not thought out properly and thoroughly, the changes could break other system components.

IQ 24: Are the software developers excited about changes to the system when the architecture is changed?

All of the interview participants stated that software developers where not excited about changes to the architecture. 50% of survey respondents agreed that software developers where not excited about changes to the architecture, while 31.25% of survey respondents did not have an opinion on this question. The impact identified for the above research question is:

- Boredom

Boredom has been identified as an impact because the participants stated software developers do not get excited about changes to the Enterprise Architecture, which include new systems development and system changes.

IQ 25 asked interview participants and survey respondents the following:

> *Do changes to the Enterprise Architecture affect software developers? If yes, how?*

All of the interview participants stated *yes*. 66.67% of the survey respondents also said *yes*. Comments made were:

- “* Working Overtime * Their loosing concentration * They stop being team members * Isolate themselves - not reachable by consultants.”
- “More work to be done. Checks need to be done to ensure changes do not have any negative effects on the rest of the system.”
- “Sometimes. Depends on what is changing. Sometimes a change will have no effect, sometimes interfaces need to be updated or written from scratch.”
- “In terms of maintenance, software developers have to adjust their knowledge of the system and business rules to suit the new architecture.”
- “Dependant on the scope of work: If enterprise architects decide to "*re-use*" tailor made components, because it is already included in the system. The component usually needs to be customized to some extent in order to make it fit in the new environment.”
- “Some development standards and procedures might change. New functionality might be available for the dev to use, etc”

Section 5.4 summarised all of the responses given by the research participants and survey respondents. It also provided clarity on why these impacts were identified as having a negative impact of software developers. It also discussed the key findings of the data analysis.

5.5 Presentation of emergent themes

During content analysis a number of secondary themes emerged. Even though some of these themes are located on the periphery of the central focus of the research topic, they were considered to contribute to the richness of the study and the characterization of its context. These themes are grouped as impacts experienced by software developers. These impacts include:

- Forming of elite software developer cliques;
- Handover and knowledge transfer;
    - Performance ratings;

    - Career growth;
    - Key talent retention; as well as
- New developer’s time lines increased.

5.5.1 Forming of elite software developer cliques

This emergent theme was identified by **<u>IQ 20</u>**:

> *When training a new software developer on the Enterprise Architecture and the corresponding system, are system rules and components clear and understandable?*

One of the interview participants commented on the forming of software developer cliques, declaring that he felt left out. This was highlighted in Chapter 4 as part of the initial list of impacts: training new developers on the system causes difficulties as there is no set structure for changes, thus meaning no solid architecture exists. Training the support team to maintain the system also results in changes to support procedures, which is normally based on business rules implemented in the system i.e. the business rule changes means the system must change. Because of this knowledge gap, software developers are often called in to assist the support team with the challenges they face. This creates more pressure on the software developers who already have time constraints and are putting in overtime. Software developers also feel ‘explaining how the system works’ takes more time than them just fixing the issue themselves.

This leads to silos of expertise, as well as cliques of software developers. There is the A-team of software developers, the ones who understand everything and can fix everything. A new software developer will take years to reach the system knowledge of a software developer who has been there longer, and with the continuous change, the gap grows daily.

5.5.2 Handover and knowledge transfer

This emergent theme was identified by **<u>IQ 20</u>**:

When training a new software developer on the Enterprise Architecture and the corresponding system, are system rules and components clear and understandable?

Four of the six interview participants commented on this question, stating that they had difficulty in doing handovers and knowledge transfer. It was stated that the gap between “how the architects envisions it” and “what happens in the background” are two different things. This impact was identified in Chapter 4, as software developers who know the system cannot leave without creating a huge knowledge gap and new software developers always feel inadequate; no matter how long they work they will never be the best. A survey participant commented:

If there is no decent system documentation and literature of the system there will always be a gap in knowledge transfer.

This led to a discussion on how software developer’s performance is perceived. An interview participant stated that:

I cannot teach someone to do my work, it takes too long. But if I cannot get someone else to do the job I will never get promoted.

This lead to the following emergent themes:

- Career growth;
- Performance ratings; as well as
- Key talent retention.

Software developers who are kept in their position because of their knowledge feel constrained when it comes to career growth. Keeping a software developer in the same position will eventually cause that software developer to leave the company, causing the company to lose key talent.

5.5.3 New software developer’s time lines increased

New software developer’s time lines increase as a result of a misconception of the amount of work involved in software development. Software developers have an idea of a standard time frame to add certain functionality to the system, for instance a new button or workflow. However, if the system is full of improvised ways to make it work, this affects the estimation of time lines given for new software developers. Software developers who have been working on the system and Enterprise Architecture know of these pitfalls and to include it in the time estimation. Handover and knowledge transfer has been identified as an impact because explaining improvised ways and workarounds tend to confuse new software developers, which makes handover and knowledge transfer a difficult task. These impacts were identified during the interviews by some on the participant's responses, but are highlighted in a response received on the survey stating:

Changes that have an ***unintended*** *impact are communicated and new scope is formed.*

This indicates a pitfall that was not identified as a primary theme, but has an impact on software developers.

This section discussed the emergent themes that were identified during the data analysis. The following section provides solutions to the impacts identified through the participant observation and the primary and emergent themes of the data analysis.

5.6 Possible solutions for key findings

The purpose of this study was to investigate the impact of Enterprise Architecture management decisions on the responsibilities, work experience and attitude towards Enterprise Architecture of the software developers in a company that develops software.

Goikoetxea claims that successful Enterprise Architecture is

… all about "lining up the ducks", so that the institutional side, the business side, the engineering side and the financial side of the picture are all addressed as a necessary condition to be met prior to and during the actual construction of the Enterprise Architecture (Goikoetxea, 2007, p.403).

When we consider this quotation, there is little or no mention of the technological and software development aspects when addressing the engineering aspect of the Enterprise Architecture, specifically within a company that develops software. However, all changes that are made to the Enterprise Architecture have a ripple effect through the company and these changes will have an impact on the jobs and responsibilities of employees throughout the company (The-Open-Group, 2009, p.183; Hoque, 2002, p.87).

At present very few resources in Information Systems literature could be found that address the impact and challenges that companies, which develop software, experience when they adopt Enterprise Architecture. The purpose of this study is to identify and investigate the impact of Enterprise Architecture management level decisions on employees, specifically software developers, with regards to the responsibilities, work experience and attitude towards Enterprise Architecture in companies that develop software. The primary list of impact which was identified included:

- Software developers are sometimes only notified of a change, not given the opportunity to give advice on what might work better, could be developed faster or a more creative way of doing things
- Not all changes to the Enterprise Architecture are analyzed thoroughly and these changes then create negative occurrences later in the process
- Negative occurrences include the system falling over, data transformation errors, workflow errors, missing data, incorrect system flow, data integrity issues and system concurrency issues.
- All of these negative occurrences impact the business view of the system, which causes the system to be called unstable, not working, full of errors and bugs and untrustworthy.
- These negative feelings are transferred to the developer, resulting in the software developer being called untrustworthy, writing unstable code and being a ‘bad’ software developer.
- Negative occurrences also take time to fix, and because of the pressure being put on software developers, this results in late hours and working overtime on a daily basis.

- Training new software developers on the system causes difficulties as there is no set structure for changes, thus meaning no solid system architecture exists. Training the support team to maintain the system also results in changes to support procedures, which is normally based on business rules implemented in the system i.e. the business rule changes require that the system must change. Because of this knowledge gap, software developers are often called in to assist the support team with the challenges they face. This creates more pressure on the software developers who already have time constraints and are putting in overtime. Software developers also feel 'explaining how the system works' takes more time than them just fixing the issue themselves.

- This leads to silos of expertise, as well as cliques of developers. There is the A-team of software developers, the ones who understand everything and can fix everything. A new software developer will take years to reach the system knowledge of a software developer who has been there longer, and with the continuous change, the gap grows daily.

- This highlights two areas of risk: software developers who know the system cannot leave without creating a huge knowledge gap and new software developers always feel inadequate; no matter how long they work they will never be the best.

- Software developers also noted that some projects the *Company* required in order to do business, which was placed at the front of the development queue and *had* to be done within half the time it would normally take because it was *business critical,* was never used, or used minimally because the users were not happy with the change.

- Software developers also implemented many of *work-arounds* in order to accommodate the changes to the Enterprise Architecture. These *work-arounds* made maintenance difficult in the sense that one work-around usually broke another work-around.

These work-around methods are jokingly referred between the software developers as: “Today I *ninja’d* the code again”.

This was confirmed through the use of surveys and interviews. Section 5.4 and Section 5.5 discussed the primary and emergent themes, which provided the researcher with a list of impacts. The following list provides a summary of the impacts identified and the possible solutions for the impacts. These solutions have not been implemented and the outcome tested as this is out of the scope of this study.

The list of impacts identified is:

- Scope Creep;
- Improvisation;
- Lessening of development standards;
- Overtime;
- Design flaws;
- Quality of final product;
- Teaching new software developers is difficult;
- Boredom;
- Unused components;
- Increased maintenance;
- Forming of elite software developer cliques;
- Handover and knowledge transfer;
    - Performance ratings;
    - Career growth ;
    - Key talent retention; as well as
- New developer’s time lines increased.

When the Enterprise Architecture is defined (or changes made to the Enterprise Architecture), include the software developers in the scoping sessions in order to avoid scope creep, overtime and design flaws. Software developers should know the system well enough to point out areas that will require additional work. Deepen the level of detail in the Enterprise Architecture to include system component dependencies, this will ensure that all parties are aware of the full scope of the changes to be made, as well as impact points across systems. This will make sure enough detail is included for all parties to make informed decisions on the timelines given for work, and what changes need to be made for the Enterprise Architecture change to successfully translate into a system change. This should address the impacts of improvisation, lessening of development standards and the quality of the final product. If the changes made here easy to perceive, understand and of a detailed level, new software developers can use the Enterprise Architecture to teach themselves about the system, asking other software developers only if an element is confusing. This addresses the impacts of new software developer's time lines increasing and teaching new software developers are difficult.

This detailed architecture means everyone can see what is happening where in the system and should sort out the forming of "elite" developers who form part of the "a-team". The transfer of knowledge from this group of software to a central repository should lessen the impacts of handover and knowledge transfer, performance ratings, career growth and key talent retention. This leaves software developers with the opportunity of growth. Software developers who have the ambition to grow in their roles as software developers want to work on as many different projects as possible during their time on a certain level. The increased level of detail on the Enterprise Architecture will enable resource sharing, i.e. software developers can be used on different systems, without the increased timelines or the ensuing boredom of the software developers who constantly do the same tasks over and over again.

With an increased level of detail in the Enterprise Architecture, this will also help the support teams to maintain the system, resulting is lesser conflicts between the development and support team. Also because items are developed up to standard, the system maintenance time will decrease.

Unused components will be identified as soon as they become obsolete, because the detailed architecture will show the unused components. This will decrease development and maintenance time, and ensure system understanding is optimal for all parties involved.

5.7 Summary

The research questions addressed in this chapter were:

1. *How are the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer impacted by Enterprise Architecture management decisions?* The answer to this question was discussed in section 5.4 and 5.5.
2. *How do software developers experience the impact of Enterprise Architecture management decisions on their responsibilities, work experience and attitude towards Enterprise Architecture?* The answer to this question was discussed in section 5.4 and 5.5.

This chapter described the findings of the analysis of the transcripts from the field interviews. The transcripts were analyzed using a simple coding process (see Section 3.3.3), which was employed to uncover the themes running through data. The themes are organized into two classes: the primary themes, relating directly to the topics of the interview questions, and the emergent themes, which are relevant themes that emerged during the coding process.

Chapter 6: Contribution

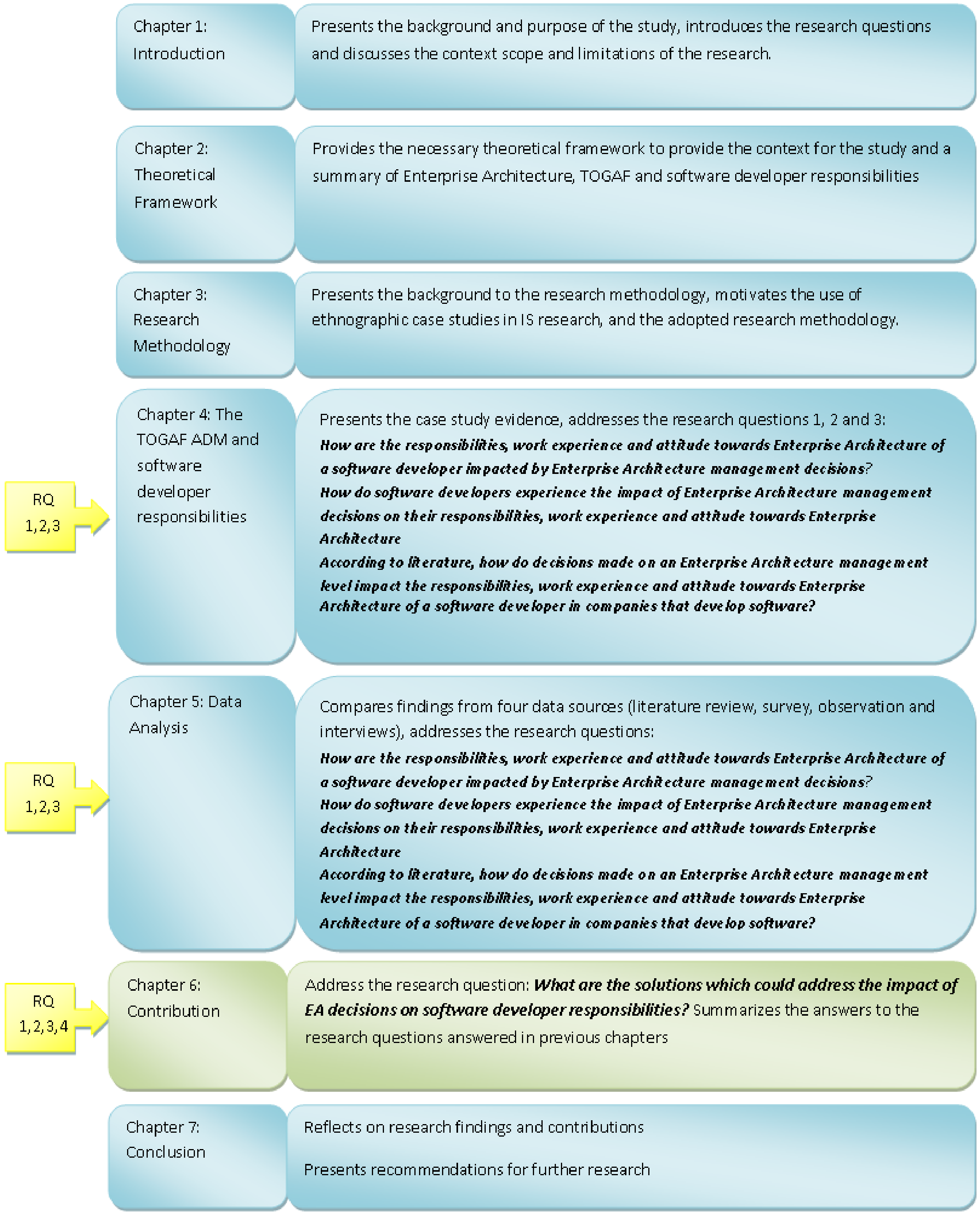

Chapter 1: Introduction
Presents the background and purpose of the study, introduces the research questions and discusses the context scope and limitations of the research.
Chapter 2: Theoretical Framework
Provides the necessary theoretical framework to provide the context for the study and a summary of Enterprise Architecture, TOGAF and software developer responsibilities
Chapter 3: Research Methodology
Presents the background to the research methodology, motivates the use of ethnographic case studies in IS research, and the adopted research methodology.
RQ 1,2,3
Chapter 4: The TOGAF ADM and software developer responsibilities
Presents the case study evidence, addresses the research questions 1, 2 and 3:
How are the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer impacted by Enterprise Architecture management decisions?
How do software developers experience the impact of Enterprise Architecture management decisions on their responsibilities, work experience and attitude towards Enterprise Architecture
According to literature, how do decisions made on an Enterprise Architecture management level impact the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer in companies that develop software?
RQ 1,2,3
Chapter 5: Data Analysis
Compares findings from four data sources (literature review, survey, observation and interviews), addresses the research questions:
How are the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer impacted by Enterprise Architecture management decisions?
How do software developers experience the impact of Enterprise Architecture management decisions on their responsibilities, work experience and attitude towards Enterprise Architecture
According to literature, how do decisions made on an Enterprise Architecture management level impact the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer in companies that develop software?
RQ 1,2,3,4
Chapter 6: Contribution
Address the research question: What are the solutions which could address the impact of EA decisions on software developer responsibilities? Summarizes the answers to the research questions answered in previous chapters
Chapter 7: Conclusion
Reflects on research findings and contributions
Presents recommendations for further research

Figure 21 Chapter 6 Dissertation Map

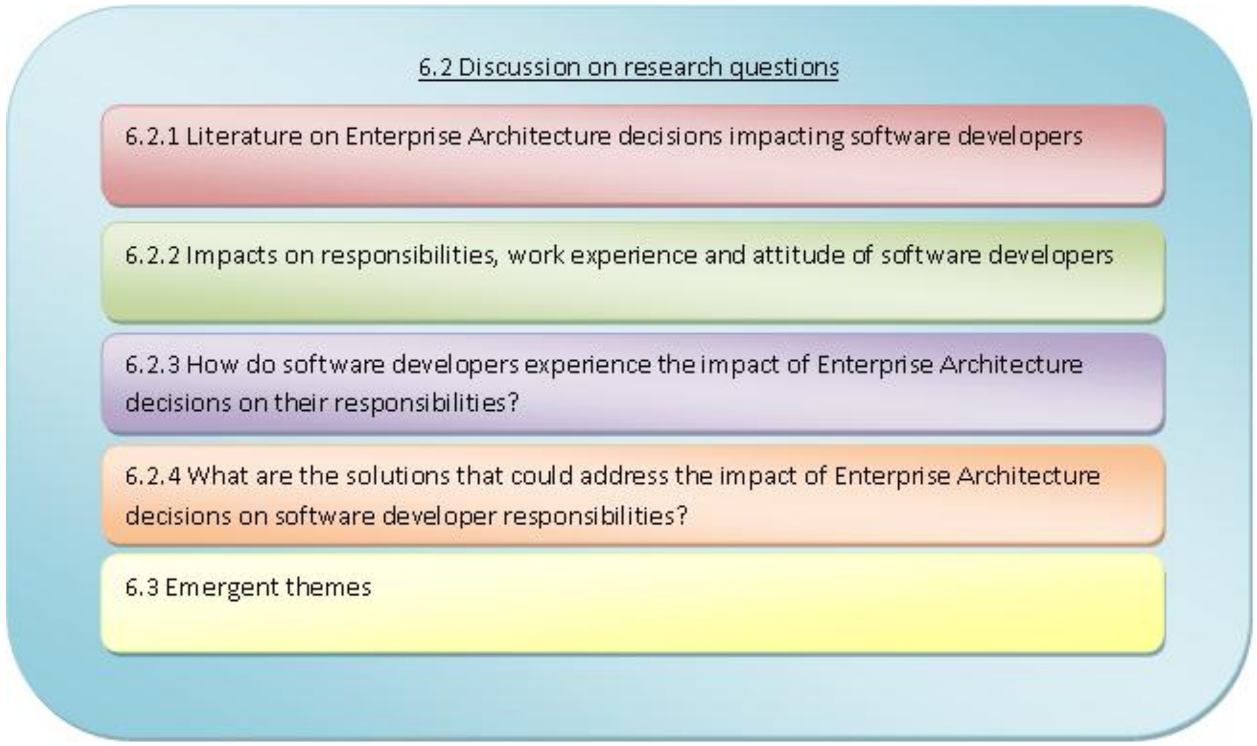

6.2 Discussion on research questions
6.2.1 Literature on Enterprise Architecture decisions impacting software developers
6.2.2 Impacts on responsibilities, work experience and attitude of software developers
6.2.3 How do software developers experience the impact of Enterprise Architecture decisions on their responsibilities?
6.2.4 What are the solutions that could address the impact of Enterprise Architecture decisions on software developer responsibilities?
6.3 Emergent themes

Figure 22 Chapter 6 Chapter Map

6.1 Introduction

The study consisted of five phases, the literature study, the contextualization, the field interviews, the survey and the participant observations. This chapter provides a discussion of the results drawn from these phases. The discussion begins with a brief summary of the study and then proceeds to address the findings relating to the four subsidiary questions and the emergent themes.

6.2 Discussion on research questions

The primary research question addressed by this study was:

How are the responsibilities, work experience and attitude of software developers impacted by the decisions made on an Enterprise Architecture management level in companies that develop software?

During the research design, the study was divided into four sub questions. The sub questions were of an exploratory nature and served to differentiate and guide the study:

1. *According to literature, how do decisions made on an Enterprise Architecture management level impact the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer in companies that develop software?* The purpose of this question was to identify whether published literature and/or solutions to the study being done existed.
2. *How are the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer impacted by Enterprise Architecture management decisions?* The purpose of this question was to see from an observer point of view, what initial impacts could be identified.
3. *How do software developers experience the impact of Enterprise Architecture management decisions on their responsibilities, work experience and attitude towards Enterprise Architecture?* The aim of this question was to obtain feedback from software developers on how they experience the impact of Enterprise Architecture management decisions on their responsibilities.
4. *What are the solutions that could address the impact of Enterprise Architecture management decisions on software developer's responsibilities?* This aim of this research question was to incorporate feedback and possible solutions from the software developers.

This research study consisted of five phases:

- Literature study;
- The contextualization;
- The field interviews;
- The survey; as well as
- The participant observations.

The researcher collected data from four sources, one of which was secondary, and three of which were primary data sources. The primary data sources included participant observation, semi structured interviews with open ended questions and a survey. The secondary data source was a literature study. During the research design, the primary research question was decomposed into four research questions of an exploratory nature. The table below shows how the four data sources contributed to the answering of the research questions.

Table 19 Relationship between Subsidiary Questions and Data Sources

RQ	***Question***	***Literature Study***	***Participant observation***	***Interview***	***Survey***
1	According to literature, how do decisions made on an Enterprise Architecture level impact the responsibilities of a software developer?	■	☐		
2	How are the responsibilities of a software developer impacted by Enterprise Architecture decisions?	☐	■	■	■
3	How do software developers experience the impact of Enterprise Architecture decisions on their responsibilities?	☐	■	■	■
4	What are the solutions that could address the impact of Enterprise Architecture decisions on software developer responsibilities?	☐	■	■	■
					☐ Denotes additional data source ■Denotes primary data source

The research findings relating to each of the subsidiary questions are discussed in Sections 6.2.1 to 6.2.3 and are followed by a discussion of the additional topics that emerged from the analysis of the transcripts of the field interviews in Section 6.3.

6.2.1 Literature on Enterprise Architecture decisions impacting software developers

The research question defined in Chapter 1 that refers to this topic is:

According to literature, how do decisions made on an Enterprise Architecture management level impact the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer in companies that develop software?

The purpose of this question was to identify if there was published literature and/or solutions to the study being done. This question was answered in Chapter 4. It was found that there are lots of sources on the *impact* of factors on Enterprise Architecture decisions, but no solid research has been done on how decisions made at Enterprise Architecture level impact the individuals and systems within an enterprise. The shortage of research on the impact of Enterprise Architecture is worrying, especially if one considers that Enterprise Architecture promises benefits with regards to the return of investment in IT.

The following table gives suggested responsibilities on an Enterprise Architecture management level in order to help minimize the impact experienced by software developers:

Table 20 Suggested Software Developer Involvement

ADM Phase	***Activity***	***Suggested SD responsibility***
Preliminary Phase: Frameworks and Principles	The organization must be prepared with the TOGAF procedures in order to facilitate a successful Enterprise Architecture project.	None
Requirements Management	The main expectation of this project is kept here. Each cycle or phase should be validated to the original expectations. Each of the phases store, prioritize, dispose or address requirements, the history is also kept here.	Software developers should work on getting the system in line with the Enterprise Architecture, so expectations should be made available to the software developers so that a bigger picture of the requirement can be

		formed. This is to facilitate system component re-use to minimize complexity
Phase A: Architecture Vision	The scope, constraints and expectations for the Enterprise Architecture project is setup and defined here. The business context is validated and the Statement of Architecture Work is created.	The full scope and possible future changes should be included for developers to see, this is so developers can plan changes and wider system impacts
Phase B: Business Architecture Phase C: Information Systems Architecture (Data and Applications) Phase D: Technology Architecture	The Enterprise Architecture is developed on the three levels: • Business • Information Systems • Technology This process involves creating a current view of the existing architecture (“as-is”) and the target architecture (“to-be”). This is then followed by a gap analysis to determine what changes should be made to the existing architecture.	When the information systems architecture is done, include the software developer who is going to work on the project specifically for data transfers, code and the “as-is” model. Scoping changes to get to the to-be model should include the developers so they can make suggestions for better solutions
Phase E: Opportunities and Solutions	When all the gaps have been identified and prioritized, these gaps are sorted into implementation plans.	Implement solutions as suggested by the software developers, not as is convenient in the architecture
Phase F: Migration Planning	A cost benefit analysis is done on all changes to be made. An Implementation Road Map is also	Software developers should be made aware of the implementation

	created noting the analyses and prioritization of the requirements	road map, as planning the development and system re-use components are key, this is to ensure that work is not duplicated and functionality unused
Phase G: Implementation Governance	Implementation Architecture Contracts are created to ensure all work is compliant with the current architecture and that work that is carried out conforms to the new architecture.	Unexpected system issues should be addressed here, for instance unforeseen changes that were not recognized or spotted in the initial design
Phase H: Architecture Change Management	Because business's vision changes, the architecture must change to accommodate the change in business. This phase ensures concurrency in the architecture	Software developers need to be aware of all changes made here to be able to evaluate potential impacts in the next architecture iteration

This section discussed the view of Literature on Enterprise Architecture decisions impacting software developers. The next section discusses the impacts on the responsibilities, work experience and attitude towards Enterprise Architecture of software developers.

6.2.2 Impacts on responsibilities, work experience and attitude of software developers

As defined in Chapter 1, the Research Question that links to the impacts that were identified was defined as:

How are the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer impacted by Enterprise Architecture management decisions?

The purpose of this question is to see from an observer point of view, what initial impacts could be identified. Chapter 4 and 5 answered this question. The list of impacts identified by software developers are:

- The Enterprise Architecture is not fully documented and causes decisions on Enterprise Architecture to be incorrect leading to incorrect system changes.
- The ownership of the data is not allocated to a specific department, and with the overlapping use of the data within different departments, the responsibility for system changes causes confusion and ambiguity, increasing the complexity of the system and confusing software developers.
- The Enterprise Architecture is not managed properly, which causes changes affecting multiple areas to the system. This is not reflected on the Enterprise Architecture, which leads to an increase in development time, makes system maintenance difficult, involves overtime, and relaxes development standards.
- Overtime on normal responsibilities - Software developers become work horses who produce code; the impact on other parts of the systems is not realized. This indicates a flaw in the Enterprise Architecture as all components on the Enterprise Architecture level are not clearly defined, as well as the best possible solutions are not being produced because the view of the software developer is not seen as part of the Enterprise Architecture.
- Component re-use - Software developers do not agree with re-using components, especially if the best practices and in-house development standards are not followed. The negatively impacts software developer because they are forced to re-use components, which means they will spend more time customizing the existing components, which is not guaranteed to fit and work the way the Enterprise Architecture envisions it, resulting in work-arounds and a lessening in the quality of the product provided.

This section discussed the impacts of Enterprise Architecture management decisions impacting software developers. The next section discusses the impacts are experienced by software developers.

6.2.3 How do software developers experience the impact of Enterprise Architecture decisions on their responsibilities?

The research question defined in Chapter 1 that refers to this topic is:

How do software developers experience the impact of Enterprise Architecture management decisions on their responsibilities, work experience and attitude towards Enterprise Architecture?

The aim of this question was to obtain feedback from software developers and how they experience the impact on their responsibilities. Chapter 4 and 5 answered this question. The list of impacts experienced by software developers include:

- Scope Creep;
- Improvisation;
- Lessening of development standards;
- Overtime;
- Design flaws;
- Quality of final product;
- Teaching new software developers is difficult;
- Boredom;
- Unused components; as well as
- Increased maintenance.

This section discussed how the impacts of Enterprise Architecture management decisions are experienced by software developers. The next section discusses possible solutions for these impacts.

6.2.4 What are the solutions that could address the impact of Enterprise Architecture decisions on software developer responsibilities?

As defined in Chapter 1, the Research Question that links to the impacts that were identified was defined as:

What are the solutions that could address the impact of Enterprise Architecture management decisions on software developer's responsibilities?

This aim of this research question was to incorporate feedback and possible solutions from the software developers. Chapter 4 and 5 answered this question. The list of possible solutions includes:

- Including software developers in the whole Enterprise Architecture life cycle if the software developer is responsible for working on the Enterprise Architecture components that will change.
- Include a greater level of detail in the Enterprise Architecture, showing system components and dependencies.
- Update the Enterprise Architecture with system changes to be included in the next Enterprise Architecture life cycle, this will ensure informed decisions by all parties.

This section discussed how the possible solutions to the impacts of Enterprise Architecture management decisions, which are experienced by software developers. The next section discusses the emergent themes for the study.

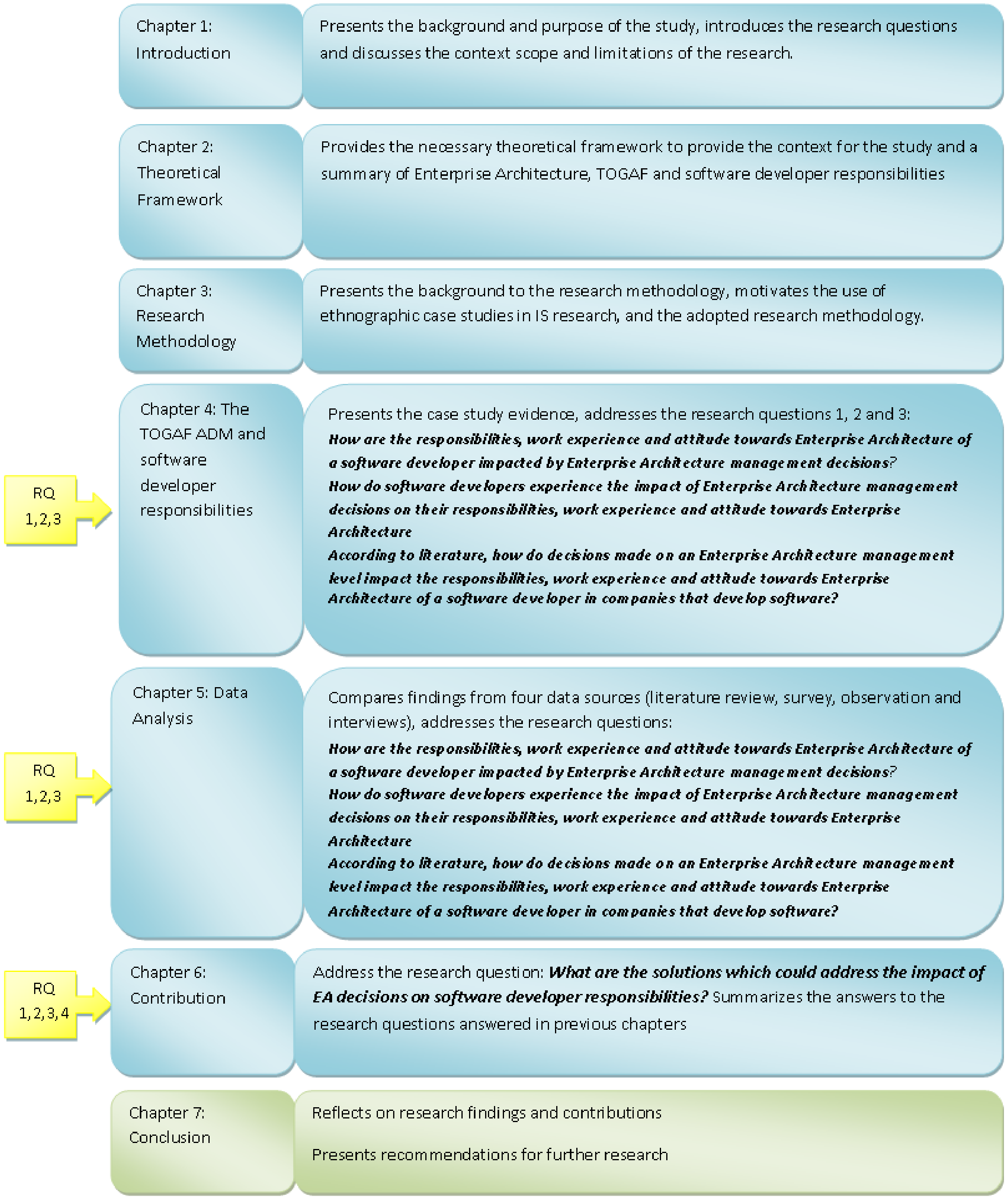
6.3 Emergent themes

In addition to the anticipated impacts relating to the interview questions, several other impacts emerged from the field interview transcripts. Those that were considered to contribute to the study include:

- Forming of elite software developer cliques;
- Handover and knowledge transfer;
    - Performance ratings;
    - Career growth ;
    - Key talent retention; as well as
- New developer's time lines increased.

Chapter 6 summarised the research findings and discussed the view of literature on the impact of Enterprise Architecture management decisions on the responsibilities, work experience and attitude towards Enterprise Architecture of software developers in companies that develop software. It also discussed how software developers experience the impacts and provided possible solutions to the impacts identified. Chapter 7 concludes the study and provides a summary of the key and emergent findings, the significance of the research, the limitations of the research and recommendations for further work.

Chapter 7: Conclusion



Chapter 1: Introduction
Presents the background and purpose of the study, introduces the research questions and discusses the context scope and limitations of the research.
Chapter 2: Theoretical Framework
Provides the necessary theoretical framework to provide the context for the study and a summary of Enterprise Architecture, TOGAF and software developer responsibilities
Chapter 3: Research Methodology
Presents the background to the research methodology, motivates the use of ethnographic case studies in IS research, and the adopted research methodology.
RQ 1,2,3
Chapter 4: The TOGAF ADM and software developer responsibilities
Presents the case study evidence, addresses the research questions 1, 2 and 3:
How are the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer impacted by Enterprise Architecture management decisions?
How do software developers experience the impact of Enterprise Architecture management decisions on their responsibilities, work experience and attitude towards Enterprise Architecture
According to literature, how do decisions made on an Enterprise Architecture management level impact the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer in companies that develop software?
RQ 1,2,3
Chapter 5: Data Analysis
Compares findings from four data sources (literature review, survey, observation and interviews), addresses the research questions:
How are the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer impacted by Enterprise Architecture management decisions?
How do software developers experience the impact of Enterprise Architecture management decisions on their responsibilities, work experience and attitude towards Enterprise Architecture
According to literature, how do decisions made on an Enterprise Architecture management level impact the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer in companies that develop software?
RQ 1,2,3,4
Chapter 6: Contribution
Address the research question: What are the solutions which could address the impact of EA decisions on software developer responsibilities? Summarizes the answers to the research questions answered in previous chapters
Chapter 7: Conclusion
Reflects on research findings and contributions
Presents recommendations for further research

Figure 23 Chapter 7 Dissertation Map

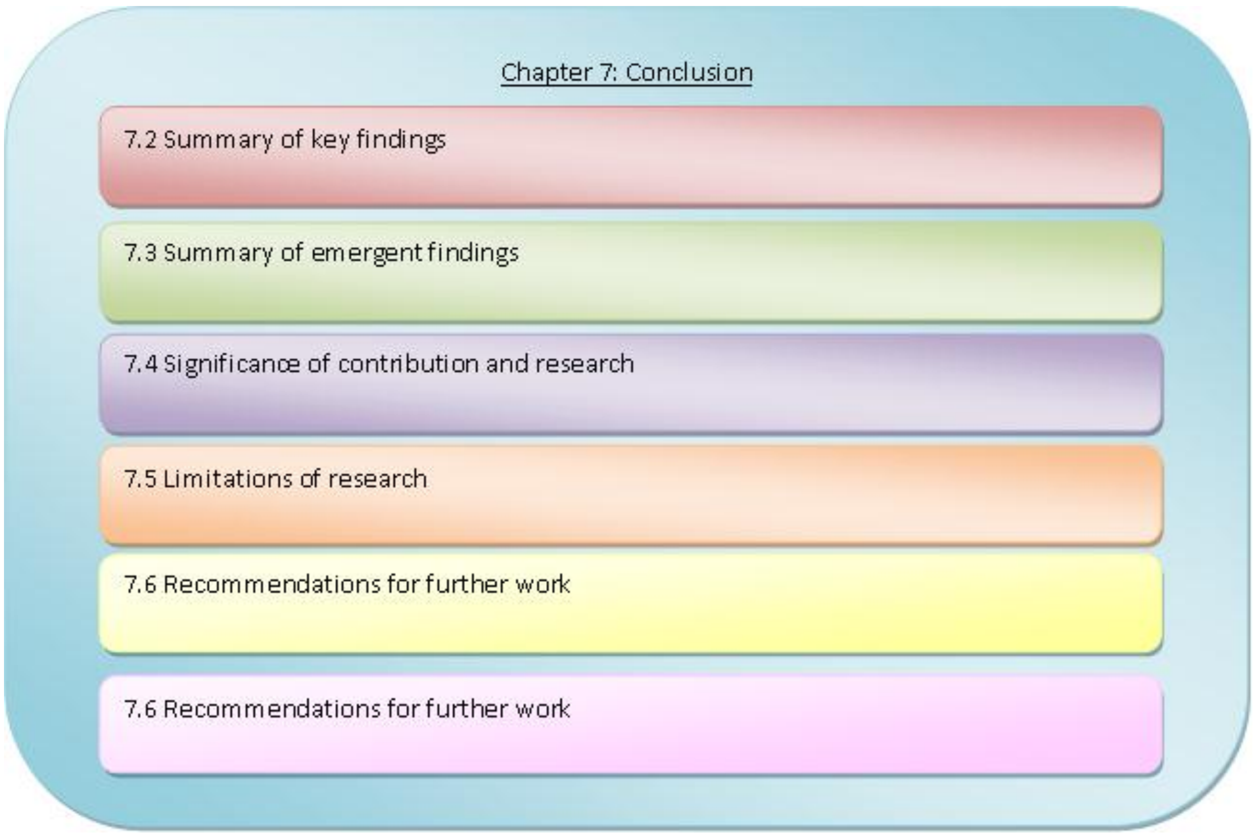

Chapter 7: Conclusion
7.2 Summary of key findings
7.3 Summary of emergent findings
7.4 Significance of contribution and research
7.5 Limitations of research
7.6 Recommendations for further work
7.6 Recommendations for further work

Figure 24 Chapter 7 Chapter Map

7.1 Introduction

With the publication of an article *A Framework for Information Systems Architecture* in the IBM Systems Journal, John Zachman pioneered the field of Enterprise Architecture (Zachman, 1987). In this article, Zachman discussed the challenges and the vision of Enterprise Architecture. According to Zachman the main challenge of enterprises in the 21st century is the management of complexity and change in increasingly distributed systems:

The cost involved and the success of the business depends increasingly on its information systems and require a disciplined approach to the management of those systems (Zachman, 1987, p.276).

Enterprise Architecture endeavours to resolve the complexity of increasingly distributed systems by aligning business vision with IT strategy, which in turn should reduce the overall costs of IT in the business and provide simpler, better and faster solutions to business problems (Zachman, 1987, p.276; Suomi et al., 2006, p.4; Ahlemann et al., 2012, p.10; Tomkowicz, 2007, p.10). To solve the challenges of managing the increased complexity of distributed systems, aligning business vision with IT strategy and reducing IT costs, Zachman (1987, p.276) stated that:

...it is necessary to use some logical construct (or architecture) for defining and controlling the interfaces and integration of all the components of the system.

Zachman (1987, p.276) further suggested that in order to facilitate the management of the logical construct:

… it likely will be necessary to develop some kind of framework for rationalizing the various architectural concepts and specifications in order to provide for clarity of professional communication, to allow for improving and integrating development methodologies and tools, and to establish credibility and confidence in the investment of systems resources.

This necessity to organise Enterprise Architecture led to the development of many Enterprise Architecture frameworks in the recent past. The main purpose of most of these frameworks is to assist with the challenges of managing the increased complexity of distributed systems, aligning business vision with IT strategy and reducing IT costs (Tambouris et al., 2011, p.150; Okunieff et al., 2011, p.58). There are many approaches for doing Enterprise Architecture (Tambouris et al., 2011, p.150, Bernard, 2012, p.109). For each of these multiple approaches there are frameworks, which are provided by different parties to serve different purposes (Tambouris et al., 2011, p.150, Okunieff et al., 2011, p.58).

An example of such an Enterprise Architecture framework is TOGAF, which has been in refinement since its creation as TAFIM in 1994 (The-Open-Group, 2009). TOGAF is at present an industry standard architecture framework (Van, 2006, p.21, The-Open-Group, 2009). It has been developed and improved since the mid-90's by IT professionals, working in The Open Group's Architecture Forum (The-Open-Group, 2009). TOGAF is a comprehensive method and set of supporting resources for

Enterprise Architecture (The-Open-Group, 2009; Meaden and Whelan, 2012, p.204; Raynard, 2008, p.59). The latest version of TOGAF is Version 9. With the help of Enterprise Architecture frameworks such as TOGAF, companies are able to design, build and evaluate an Enterprise Architecture, which is appropriate for their company (Raynard, 2008, p.59).

As TOGAF is at present one of the most adopted and cited frameworks, it is plausible to argue that a company that wishes to implement Enterprise Architecture should benefit from the use of TOGAF (Raynard, 2008, p.60; The-Open-Group, 2009; Greefhorst and Proper, 2011, p.183). This was confirmed by an investigation done by the company used in the case study within the context of this research.

The purpose of this study was to investigate the impact of Enterprise Architecture management decisions on the responsibilities, work experience and attitude towards Enterprise Architecture of the software developers in a company that develops software.

Goikoetxea claims that successful Enterprise Architecture is

… all about "lining up the ducks", so that the institutional side, the business side, the engineering side and the financial side of the picture are all addressed as a necessary condition to be met prior to and during the actual construction of the Enterprise Architecture (Goikoetxea, 2007, p.403)

When we consider this quotation, there is little or no mention of the technological and software development aspects when addressing the engineering aspect of the Enterprise Architecture, specifically within a company that develops software. However, all changes that are made to the Enterprise Architecture have a ripple effect through the company and these changes will have an impact on the jobs and responsibilities of employees throughout the company (The-Open-Group, 2009 p.183; Hoque, 2002, p.87).

At present very few resources in Information Systems literature could be found that address the impact and challenges which companies who develop software experience when they adopt Enterprise Architecture. The purpose of this study was to identify and investigate the impact of Enterprise Architecture management level decisions on employees, specifically software developers, with regards to the responsibilities, work experience and attitude towards Enterprise Architecture in companies that develop software.

During the implementation of a company's Enterprise Architecture numerous technical and organizational issues need to be addressed (Saha, 2009, p.27; Andersen et al., 2011, p.27). These challenges will naturally differ according to the environment or context of the Enterprise Architecture implementation. The purpose of this study was to investigate the impact of Enterprise Architecture decisions on the responsibilities, work experience and attitude towards Enterprise Architecture of software developers in companies that develop software.

The primary research question addressed by this study was:

> *How are the responsibilities, work experience and attitude of software developers impacted by the decisions made on an Enterprise Architecture management level in companies that develop software?*

During the research design, the study was divided into four sub questions. The sub questions were of an exploratory nature and served to differentiate and guide the study:

1. *According to literature, how do decisions made on an Enterprise Architecture management level impact the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer in companies that develop software?* The purpose of this question was to identify whether published literature and/or solutions to the study being done existed.
2. *How are the responsibilities, work experience and attitude towards Enterprise Architecture of a software developer impacted by Enterprise Architecture management decisions?* The purpose of this question was to see from an observer point of view, what initial impact could be identified.
3. *How do software developers experience the impact of Enterprise Architecture management decisions on their responsibilities, work experience and attitude towards Enterprise Architecture?* The aim of this question was to obtain feedback

from software developers on how they experience the impact of Enterprise Architecture management decisions on their responsibilities.

4. *What are the solutions that could address the impact of Enterprise Architecture management decisions on software developer's responsibilities?* This aim of this research question was to incorporate feedback and possible solutions from the software developers.

Since the main focus of a research study is to discover answers that will help explain and achieve the aim and objectives of the planned research, a methodological approach is required to facilitate the research process (Maykut, 1994, p.43; Mitchell and Jolley, 2009, p.53; Munizzo and Musial, 2010, p. 30). The strategy adopted in this study was an interpretive qualitative ethnographic case study, which was supplemented by surveys (Cohen et al., 2007, p.255; Wiebe et al., 2009, p. 597; Simons, 2009, p. 22; Lee et al., 1997, p. 278).

The initial observation of a problem was made from informal discussions with fellow colleagues. This was followed by a primary list of impacts that was gathered through the use of participant observation. During this exercise the researcher gained insight through conversations with other software developers in the case study environment (Spindler and Hammond, 2006, p.34; DeWalt and DeWalt, 2010, p.1; Angrosino, 2008, p.4; Murchison, 2010, p.7). After the initial lists of issues were compiled, a detailed literature study was executed to determine whether literature addresses the impact on employees of a company that executes an Enterprise Architecture implementation. This study provided the theoretical underpinning, which enabled the researcher to establish a thorough background on Enterprise Architecture, TOGAF, Software development and the Software Development Life Cycle (SDLC). The theoretical framework provided the contextual background for the study as well as the basis for the analysis of the findings of the study (Cassell and Symon, 2004, p. 324; Farquhar, 2012, p. 37; Runeson et al., 2012; Wiebe et al., 2009, p.813). The study also focused on the specific responsibilities of software developers within the phases of the SDLC, which is included in the theoretical framework of this study. Because of the lack of literature on research with regards to the impact of Enterprise Architecture management decisions on the responsibilities, work experience and attitude towards Enterprise Architecture of software developers in companies that develop software during the initial investigation, this study was motivated and executed.

After the literature review was conducted, the list of initial impacts identified through researcher participant observation was structured into interview questions. The interviews were conducted as semi-structured field interviews using open-ended questions (Klandermans and Staggenborg, 2002, p.93; Flick, 2009, p.165; Remenyi, 2011, p.20). The interview questions were designed to allow the participants to agree or disagree with the findings of the participant observation, as well as to give room in the study for the experiences of other software developers. These interviews also helped to determine the attitude of the software developers towards Enterprise Architecture. The software developers were also asked to comment on the impacts of Enterprise Architecture management decisions on their responsibilities.

The impacts obtained from these interviews were compared to the initial list, which was obtained from the participant observation, and impacts were either confirmed or updated with new information. In some instances, the descriptions of the impact were altered to generalize or group the different impacts accordingly (Wiebe et al., 2009, p.474; Outhwaite and Turner, 2007, p. 107).

This list of impacts was converted into a survey comprising of yes/no questions. The survey asked software developers whether they agreed with a specific impact or not. This survey was distributed to software developers in other companies who adopted Enterprise Architecture. The purpose of the survey was to confirm the ethnographic case study results. In addition to the confirmation of the results, the survey allowed the researcher to uncover different facets to the study and enabled an investigation through an appropriate combination of methods and sources (Marschan-Piekkari and Welch, 2004, p.129; Brown, 2008, p. 221).

This confirmed list of impacts and possible solutions were compiled and formalized as the contribution of the study. Section 7.2 presents a summary of the key findings of this study.

7.2 Summary of key findings

This study found that decisions made on an Enterprise Architecture level have negative impacts on the responsibilities of software developers. Impacts identified are:

- The Enterprise Architecture is not fully documented and causes decisions on Enterprise Architecture to be incorrect leading to incorrect system changes;
- The ownership of the data is not allocated to a specific department;
- The Enterprise Architecture is not managed properly;
- Scope Creep;
- Improvisation;
- Lessening of development standards;
- Overtime;
- Design flaws;
- Quality of final product;
- Teaching new software developers is difficult;
- Boredom;
- Unused components; as well as
- Increased maintenance.

7.3 Summary of emergent findings

This study found that decisions made on an Enterprise Architecture level have negative impacts on the responsibilities of software developers. Emergent themes identified are:

- Forming of elite software developer cliques;
- Handover and knowledge transfer;
  - Performance ratings;
  - Career growth ;
  - Key talent retention; as well as
- New developer’s time lines increased.

The study also identified a list of possible solutions for the impacts identified. The list of possible solutions includes:

- Including software developers in the whole Enterprise Architecture life cycle if the software developer is responsible for working on the Enterprise Architecture components that will change.
- Include a greater level of detail in the Enterprise Architecture, showing system components and dependencies.
- Update the Enterprise Architecture with system changes to be included in the next Enterprise Architecture life cycle, this will ensure informed decisions by all parties.

7.4 Significance of contribution and research

In Section 1.2: purpose of the study, the contribution and importance of this study was discussed. The results of the study indicate that the research findings are important and can contribute to the community within Information Systems research, particularly for researchers whose area of research expertise focuses on Enterprise Architecture implementation and software development. The research findings could also be of interest to Enterprise Architecture system vendors that wish to capture the enterprise market which deals with in-house development teams. The findings from this study are important to the academic body of knowledge in that:

- The findings offer exploratory insight into the impact of Enterprise Architecture management decisions on the responsibilities of software developers.
- The findings offer an understanding of software developers experience these impacts
- The findings offer an understanding of the requirements that need to met by the Enterprise Architecture so that decisions made on an Enterprise Architecture management level do not impact software developers negatively
- The findings from this study could aid in the conceptualization of new, as well as extended and improved, models and frameworks of Enterprise Architecture systems for use by companies who develop software in-house.
- This study serves as a basis for further research initiatives in different industries, in both developing and developed economies, to stimulate in-depth inquiry.

7.5 Limitations of research

Research will always have some shortcomings (Yin, 2009). Each research strategy can lead to unique results, depending on how the research methodology was pursued. If the chosen research methodology were to be carried out under different circumstances, different results would most probably emerge. The limitations of the research methodology adopted in this study are as follows:

One of the challenges of survey research strategy is generalisability (Oates, 2006). The number of participating software developers was limited. The number of participants could have included more sample subjects from different software development. This would have provided more representativeness and generalisability of findings across software development sectors.

The timeline for carrying out this survey could have been longitudinal, which would have allowed for the measurement of impacts of Enterprise Architecture decisions on the responsibilities of software developers over a period of time, taking into consideration the maturity of the company's Enterprise Architecture. The longitudinal study would presumably provide further insight and distinctive comparisons of findings. The scope and length of a Master's study does not, however, make it feasible for a longitudinal study to be done.

7.6 Recommendations for further work

As indicated in Chapter 1, it was anticipated that the results of this exploratory study could serve as a basis for further comparative investigations in various companies and in various core economic industries. Further in-depth, cross-sector and inter-industry studies should reveal additional results and expand on current results, thus providing a detailed analysis of solutions to the impact of Enterprise Architecture decisions of software developer's responsibilities that need to be considered. A number of recommendations for future work can be made, i.e.:

- Refine and improve the list of impacts;
- Correlate the proposed solutions to the impacts experienced;
- Validate the proposed solutions;

- Conduct case studies on the application of proposed solutions;
- Match individual impacts to enterprise impacts;
- Analyze various moderating effects on the impacts experienced;
- Use of alternative research strategies, such as action research and usability experiments.

The possible solutions proposed can be refined by considering issues as diverse as the complexities of inter-organizational relationships, gender differentials of software development teams and Enterprise Architecture maturity, the Enterprise Architecture framework used and issues associated with adaptability, expandability and ease of access across multi-faceted platforms. The emphasis of this study was on the impacts experienced by software developers by decisions made on an Enterprise Architecture level and, as such, the implementation of the possible solutions was only briefly explored. A recommendation is to conduct research correlating these possible solutions and the outcomes of the implementation to the impacts experienced.

It is suggested that further validation would improve the accuracy of the holistic understanding required in terms of the possible solutions provided by this study. It is all very well to understand the impacts and possible solutions and translating this to models, frameworks, guidelines, policies. However, the real benefit is derived when this is applied to the real-world context. Real-world cases will support our understanding and reiterate the importance of the possible solutions provided.

7.7 Concluding statement

The purpose of this study was to investigate the impacts of Enterprise Architecture decisions on software developer’s responsibilities, work experience and attitude towards Enterprise Architecture of software developers in companies that develop software, and provide possible solutions for the impacts experienced.

The theoretical framework provided background information on the topics studied. Elements from participant observation were used as the basis of interview questions used to determine how software developers experience the impact of Enterprise Architecture decisions. Although there was general agreement that Enterprise

Architecture negatively impacts the responsibilities of software developers, the principal concern was to provide possible solutions to these negative impacts.

It was found that there are lots of sources on the *impact* of factors on Enterprise Architecture decisions, but no solid research has been done on how decisions made at Enterprise Architecture level impact the individuals and systems within an enterprise. The shortage of research on the impact of Enterprise Architecture is worrying, especially if one considers that Enterprise Architecture promises benefits with regards to the return of investment in IT.

The list of impacts identified by software developers are include from the Enterprise Architecture is not fully documented and causes decisions on Enterprise Architecture to be incorrect leading to incorrect system changes to a more relevant topic such as increased maintenance.

Although Enterprise Architecture promises to align business vision with IT strategy, reducing the overall costs of IT in the business and providing simpler, better and faster solutions to business problems, this study has shown that decisions made on an Enterprise Architecture level negatively impacts the responsibilities, work experience and attitude towards Enterprise Architecture of software developers.

Following the tradition of interpretive research, the study revealed useful findings beyond those originally expected.

These themes are grouped as impacts experienced by software developers. These impacts include the forming of elite software developer cliques, as well as new developer’s time lines increased.

The majority of these additional findings related to problems in the *Company* that are attributable to its software development environment and the Enterprise Architecture used.

The problems relating to the impacts identified concern important issues such as the management of intellectual property, career development and long-term product support. These issues need to be addressed by the *Company* irrespective of whether a specific Enterprise Architecture framework is adopted.

Although the field interviews addressed some of the potential problems that the *Company* might experience with the impacts of Enterprise Architecture decisions on the responsibilities, work experience and attitude towards Enterprise Architecture of software developers, additional potential problems were also identified by participants. These related largely to impacts that were not identified during the participant observation. Although, the field interviews identified certain perceived problems with the impacts of Enterprise Architecture management decisions on the responsibilities of software developers, they also provided possible solutions to the impacts experienced. The list of possible solutions includes:

- Including software developers in the whole Enterprise Architecture life cycle if the software developer is responsible for working on the Enterprise Architecture components that will change.
- Include a greater level of detail in the Enterprise Architecture, showing system components and dependencies.
- Update the Enterprise Architecture with system changes to be included in the next Enterprise Architecture life cycle, this will ensure informed decisions by all parties.

In conclusion, the impact of Enterprise Architecture decisions on the responsibilities of software developers were investigated in companies that develop software. In this respect, the study identified impacts of Enterprise Architecture management decisions as well as possible solutions to these impacts.

List of references

ABRAHAMSSON, P., MARCHESI, M. and MAURER, F. 2009. *Agile Processes in Software Engineering and Extreme Programming: 10th International Conference, XP 2009, Pula, Sardinia, Italy, May 25-29, 2009, Proceedings*, Springer.

ABRAMOWICZ, W., TOLKSDORF, R. and WECEL, K. 2010. *Business Information Systems Workshops: BIS 2010 International Workshop, Berlin, Germany, May 3-5, 2010, Revised Papers*, Springer.

AHLEMANN, F., STETTINER, E., MESSERSCHMIDT, M. and LEGNER, C. 2012. *Strategic Enterprise Architecture Management: Challenges, Best Practices, and Future Developments*, Springer.

AIGUIER, M., BRETAUDEAU, F. and KROB, D. 2010. *Complex Systems Design and Management: Proceedings of the First International Conference on Complex Systems Design and Management CSDM 2010*, Springer.

ANDERSEN, K. N., FRANCESCONI, E., GRÖNLUND, A. and VAN ENGERS, T. M. 2011. *Electronic Government and the Information Systems Perspective: Second International Conference, EGOVIS 2011, Toulouse, France, August 29 -- September 2, 2011, Proceedings*, Springer.

ANGROSINO, M. 2008. *Doing Ethnographic and Observational Research*, SAGE Publications.

AVISON, D. E. and PRIES-HEJE, J. 2005. *Research In Information Systems: A Handbook For Research Supervisors And Their Students*, Elsevier/Butterworth-Heinemann.

BEKER, I. 2011. *Proceedings of the XV International Scientific Conference on Industrial Systems (IS'11)*, Publisher.

BENTE, S., BOMBOSCH, U. and LANGADE, S. 2012. *Collaborative Enterprise Architecture: Enriching EA With Lean, Agile, and Enterprise 2.0 Practices*, Elsevier Science and Technology.

BERNARD, S. A. 2012. *An Introduction to Enterprise Architecture: Third Edition*, AuthorHouse.

BLAIKIE, N. 2009. *Designing Social Research*, Polity.

BLEVINS, T., HARRISON, R. and HOMAN, P. 2007. *Togaf Version 8.1.1 Enterprise Edition: A Pocket Guide*, Van Haren Publishing.

BOGUE, R. 2004. *The Many Faces of a Developer* [Online]. Available: http://www.developer.com/design/article.php/3439651/The-Many-Faces-of-a-Developer.htm [Accessed November 23, .

BOGUE, R. 2005. *Anatomy of a Software Development Role: Developer* [Online]. Available: http://www.developer.com/java/other/article.php/3511566/Anatomy-of-a-Software-Development-Role-Developer [Accessed June 9, .

BRAY, I. 2012. *Healthy Employees, Healthy Business: Easy, Affordable Ways to Promote Workplace Wellness*, Nolo.

BRENNAN, K. 2009. *A Guide to the Business Analysis Body of Knowledge (Babok Guide)*, International Institute of Business Analysis.

BROWN, A. 2008. *7th European Conference on Research Methodology for Business and Management Studies: Ecrm 2008*, Academic Conferences Limited.

BRYMAN, A. and BELL, E. 2007. *Business Research Methods*, Oxford University Press, USA.

BUCHANAN, D. A. and BRYMAN, A. 2009. *The Sage Handbook of Organizational Research Methods*, Sage.

BUDDENBAUM, J. M. and NOVAK, K. B. 2001. *Applied communication research*, Iowa State University Press.

BURNETT, K. 1998. *The Project Management Paradigm*, Springer.

CAMARINHA-MATOS, L. M., PEREIRA, P. and RIBEIRO, L. 2010. *Emerging Trends in Technological Innovation: First Ifip Wg 5.5/Socolnet Doctoral Conference on Computing,*

Electrical and Industrial Systems, Doceis 2010, Costa de Caparica, Portugal, February 22-24, 2010, Proceedings, Springer.
CAMP, O. 2004. *Enterprise Information Systems V*, Kluwer Academic.
CAMPBELL, A. R. G. 2007. *A definition of Enterprise Architecture* [Online]. Available: http://ingenia.wordpress.com/a-definition-of-enterprise-architecture/.
CARPENTER, H. W. and ARIZONA STATE, U. 2008. *The development of a conceptual framework to understand statewide articulation system design*, Arizona State University.
CARROLL, S. and DAUGHTREY, T. 2007. *Fundamental Concepts for the Software Quality Engineer*, ASQ Quality Press.
CASSELL, C. and SYMON, G. 2004. *Essential Guide to Qualitative Methods in Organizational Research*, SAGE Publications.
CATER-STEEL, A. and AL-HAKIM, L. 2009. *Information Systems Research Methods, Epistemology, And Applications*, Information Science Reference.
COHEN, L., MANION, L., MORRISON, K. And MORRISON, K. R. B. 2007. *Research methods in education*, Routledge.
DAN, A., GITTLER, F. and TOUMANI, F. 2010. *Service-Oriented Computing. ICSOC/ServiceWave 2009 Workshops: International Workshops, ICSOC/ServiceWave 2009, Stockholm, Sweden, November 23-27, 2009, Revised Selected Papers*, Springer.
DANE, F. C. 2010. *Evaluating Research: Methodology for People Who Need to Read Research*, SAGE Publications.
DESAI, K. C. 2009. *GREAT IDEAS TO Boost Your Buisness*, Sterling Publishers Pvt. Ltd.
DEWALT, K. M. and DEWALT, B. R. 2010. *Participant Observation: A Guide for Fieldworkers*, AltaMira Press.
DICTIONARY, O. O. 2012. *Oxford Online Dictionary* [Online]. Available: http://oxforddictionaries.com/definition/empirical.
DOOLEY, J. 2011. *Software Development and Professional Practice*, Apress.
DOOM, C. 2010. *An Introduction to Business Information Management*, ASP-VUB Press.
DUBEY, S. S. 2011. *It Strategy and Management*, PHI Learning.
DWIVEDI, Y. K., WADE, M. R. and SCHNEBERGER, S. L. 2011. *Information Systems Theory: Explaining and Predicting Our Digital Society*, Springer.
ELLIS, H. J. C., DEMURJIAN, S. A. and NAVEDA, J. F. 2009. *Software Engineering: Effective Teaching and Learning Approaches and Practices*, Information Science Reference.
ERBSCHLOE, M. 2003. *Socially Responsible It Management*, Digital Press.
FARQUHAR, J. D. 2012. *Case Study Research for Business*, SAGE Publications.
FETTERMAN, D. M. 2010. *Ethnography: step-by-step*, SAGE.
FIELD, M., KELLER, L. and UNIVERSITY, O. 1998. *Project Management*, International Thomson Business Press.
FILIP, J., CORDEIRO, J. and CARDOSO, J. 2008. *Enterprise Information Systems: 9Th International Conference, ICEIS 2007, Funchal, Madeira, June 12-16, 2007, Revised Selected Papers*, Springer London, Limited.
FISHER, I. K. 2004. *How to Start a Career in Information Technology*, Ian K. Fisher.
FLICK, U. 2009. *An Introduction to Qualitative Research*, SAGE Publications.
GIACHETTI, R. E. 2010. *Design of Enterprise Systems: Theory, Architecture, and Methods*, Taylor and Francis Group.
GILL, T. G. 2011. *Informing with the Case Method: A Guide to Case Method Research, Writing, and Facilitation*, Informing Science.
GLINER, J. A. and MORGAN, G. A. 2000. *Research methods in applied settings: an integrated approach to design and analysis*, Lawrence Erlbaum.
GLOBAL, I. and STAFF, I. R. M. A. 2010. *Enterprise Information Systems: Concepts, Methodologies, Tools and Applications*, IGI Global.

GOIKOETXEA, A. 2007. *Enterprise Architectures and Digital Administration: Planning, Design and Assessment*, World Scientific.

GRADY, J. O. 2006. *System Requirements Analysis*, Elsevier Academic Press.

GREEFHORST, D. and PROPER, E. 2011. *Architecture Principles: The Cornerstones of Enterprise Architecture*, Springer.

GREEN, A., STANKOSKY, M. and VANDERGRIFF, L. J. 2010. *In Search of Knowledge Management: Pursuing Primary Principles*, Emerald.

GUTBROD, R. and WIELE, C. 2012. *The Software Dilemma: Balancing Creativity and Control on the Path to Sustainable Software*, Springer.

HALPIN, T., KROGSTIE, J., NURCAN, S., PROPER, E., SCHMIDT, R., SOFFER, P. and UKOR, R. 2009. *Enterprise, Business-Process and Information Systems Modeling: 10th International Workshop, BPMDS 2009, and 14th International Conference, EMMSAD 2009, Held at CAiSE 2009, Amsterdam, the Netherlands, June 8-9, 2009, Proceedings*, Springer.

HAMILTON, M. 1999. *Software Development: Building Reliable Systems*, Prentice Hall.

HAMMAMI, O., KROB, D. and VOIRIN, J. L. 2012. *Complex Systems Design and Management: Proceedings of the Second International Conference on Complex Systems Design and Management CSDM 2011*, Springer.

HAMMERSLEY, M. and ATKINSON, P. 2007. *Ethnography: principles in practice*, Routledge.

HANSCHKE, I. 2010. *Strategic IT Management*, Springer Berlin Heidelberg.

HASS, K. B. 2007. *The Business Analyst As Strategist: Translating Business Strategies Into Valuable Solutions*, Management Concepts.

HAUSMAN, K. and COOK, S. 2010. *IT Architecture For Dummies*, John Wiley and Sons.

HENDERSON, H. 2009. *Encyclopedia of Computer Science and Technology*, Facts On File, Incorporated.

HENTZEN, W. and NOWAK, P. 2002. *The Software Developer's Guide*, Hentzenwerke Publishing.

HINKEL, E. 2005. *Handbook Of Research In Second Language Teaching And Learning*, L. Erlbaum Associates.

HOQUE, F. 2002. *The Alignment Effect: How to Get Real Business Value Out of Technology*, Financial Times/Prentice Hall PTR.

HORCH, J. W. 2003. *Practical Guide to Software Quality Management*, Artech House.

HOWCROFT, D. and TRAUTH, E. M. 2005. *Handbook of Critical Information Systems Research: Theory And Application*, Edward Elgar.

IAN, S. 2009. *Online Banking and the Role of CRM: The Impact of the Internet as Online Business Platform on CRM (Study of Online Banking in the UK)*, GRIN Verlag GmbH.

ISACA 2011. *CGEIT Review Manual 2011*, Information Systems Audit and Control Association.

JAAP SCHEKKERMAN, I. F. E. A. D. 2006. *How to survive in the jungle of enterprise architecture framework: creating or choosing an enterprise architecture framework*, Trafford Publishing.

JAMSHIDI, M. 2011. *System of Systems Engineering: Innovations for the Twenty-First Century*, Wiley.

JOHANNESSON, P., KROGSTIE, J. and OPDAHL, A. L. 2012. *The Practice of Enterprise Modeling: 4th IFIP WG 8.1 Working Conference, PoEM 2011 Oslo, Norway, November 2-3, 2011 Proceedings*, Springer.

KAHIN, B. and ABBATE, J. 1995. *Standards Policy for Information Infrastructure*, MIT Press.

KAN, S. H. 2003. *Metrics and Models in Software Quality Engineering*, Addison-Wesley.

KELLY, A. 2008. *Changing Software Development: Learning to Become Agile*, Wiley.

KHOSROW-POUR, M. 2006. Emerging Trends and Challenges in Information Technology Management: 2006 Information Resources Management Association International

Conference, Washington, DC, USA, May 21-24, 2006. *Information Resources Management Association. International Conference.* IDEA Group Pub.

KHOSROWPOUR, M. 2005. *Encyclopedia of Information Science and Technology*, Idea Group Reference.

KING, J. L. and LYYTINEN, K. 2006. *Information Systems: The State of the Field*, J. Wiley and Sons.

KIRIKOVA, M. 2002. *Information Systems Development: Advances in Methodologies, Components, and Management*, Kluwer Academic/Plenum Publishers.

KIYOKI, Y. 2006. *Information Modelling And Knowledge Bases XVII*, IOS Press.

KLANDERMANS, B. and STAGGENBORG, S. 2002. *Methods Of Social Movement*, University of Minnesota Press.

KOSSIAKOFF, A., SWEET, W. N., SEYMOUR, S. and BIEMER, S. M. 2010. *Systems Engineering Principles and Practice*, John Wiley and Sons.

KOTHARI, D. C. R. 2008. *Research methodology: methods and techniques*, New Age International (P) Ltd.

KUMAR, R. 2005. *Research methodology: a step-by-step guide for beginners*, SAGE.

LAAR, P. V. D. and PUNTER, T. *Views on Evolvability of Embedded Systems*, Springer.

LAND, M. O., PROPER, E., WAAGE, M., CLOO, J. and STEGHUIS, C. 2008. *Enterprise Architecture: Creating Value by Informed Governance*, Springer.

LANGER, A. M. 2007. *Analysis and Design of Information Systems*, Springer.

LANKHORST, M. 2005. *Enterprise architecture at work: modelling, communication, and analysis*, Springer.

LAPALME, J. 2011. 3 Schools of Enterprise Architecture. *IEEE,* Preprint.

LEE, A., LIEBENAU, J., DEGROSS, J. I. and PROCESSING, I. F. F. I. 1997. *Information Systems and Qualitative Research*, Chapman and Hall.

LEE, R. 2011. *Computers, Networks, Systems, and Industrial Engineering 2011*, Springer.

LEFFINGWELL, D. 2010. *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise*, Addison-Wesley.

LEWIS, J. 2008. *Sdlc 100 Success Secrets - Software Development Life Cycle (Sdlc) 100 Most Asked Questions, Sdlc Methodologies, Tools, Process and Business Models*, Emereo Pty Limited.

LILLEHAGEN, F. and KROGSTIE, J. 2008. *Active Knowledge Modeling of Enterprises*, Springer.

LOCKSTROM, M. 2007. *Low-Cost Country Sourcing*, DUV.

MAHMOOD, Z. and HILL, R. 2011. *Cloud Computing for Enterprise Architectures*, Springer.

MARSCHAN-PIEKKARI, R. and WELCH, C. 2004. *Handbook of Qualitative Research Methods for International Business*, Edward Elgar.

MAYKUT, P. 1994. *Beginning Qualitative Research: A Philosophical and Practical Guide*, Taylor and Francis.

MCCONNELL, S. 2009. *Software Project Survival Guide*, Microsoft Press.

MEADEN, G. and WHELAN, J. 2012. *Business Architecture: A Practical Guide*, Ashgate Publishing Company.

MEYERS, K. 1999. A set of Principles for conducting and evaluating interoretive field studies in Information systems. *MIS Quarterly,* 23**,** 67-94.

MILLS, A. J., DUREPOS, G. and WIEBE, E. *Encyclopedia of case study research*, SAGE.

MITCHELL, M. L. and JOLLEY, J. M. 2009. *Research Design Explained*, Wadsworth.

MITTRA, S. S. 2002. *Database Performance Tuning and Optimization: Using Oracle*, Springer.

MOHAPATRA, S. and SINGH, R. P. 2012. *Information Strategy Design and Practices*, Springer.

MOLLER, C. and CHAUDHRY, S. 2012. *Advances in Enterprise Information Systems II*, Taylor and Francis.

MORLEY, D. and PARKER, C. S. 2009. *Understanding Computers: Today and Tomorrow, Comprehensive*, Course Technology.

MOURATIDIS, H. and GIORGINI, P. 2006. *Integrating Security and Software Engineering: Advances and Future Visions*, Igi Global.

MUNIZZO, M. A. and MUSIAL, L. V. 2010. *General Report Writing and Case Studies*, Cengage Learning.

MURCHISON, J. 2010. *Ethnography Essentials: Designing, Conducting, and Presenting Your Research*, John Wiley and Sons.

MYKITYSHYN, M. G. 2007. *Assessing the Maturity of Information Architectures for Complex Dynamic Enterprise Systems,* Georgia Institute of Technology, Georgia Institute of Technology.

O'CONNOR, R., ROUT, T., MCCAFFERY, F. and DORLING, A. 2011. *Software Process Improvement and Capability Determination: 11th International Conference, SPICE 2011, Dublin, Ireland, May 30 – June 1, 2011. Proceedings*, Springer.

O'REILLY, K. 2008. *Key Concepts in Ethnography*, SAGE Publications.

OATES, B. J. 2006. *Researching information systems and computing*, SAGE.

OKUNIEFF, P. E., BOARD, N. R. C. T. R., PROGRAM, T. C. R., ADMINISTRATION, U. S. F. T. and CORPORATION, T. D. 2011. *Transit Enterprise Architecture and Planning Framework*, Transportation Research Board.

OLSEN, W. 2011. *Data Collection: Key Debates and Methods in Social Research*, SAGE Publications.

OUTHWAITE, W. and TURNER, S. 2007. *The SAGE Handbook of Social Science Methodology*, SAGE Publications.

OXFORD 2009. Oxford Online Dictionary. Oxford.

PATTON, M. Q. 2002. *Qualitative Research and Evaluation Methods*, Sage.

PERKS, C. and BEVERIDGE, T. 2002. *Guide to Enterprise IT Architecture*, Springer.

PERRY, G. M. 2002. *Absolute Beginner's Guide to Programming*, Que Pub.

POTTER, W. J. 1996. *An analysis of thinking and research about qualitative methods*, L. Erlbaum Associates.

PROPER, E., HARMSEN, F. and DIETZ, J. L. G. 2009. *Advances in Enterprise Engineering II: First NAF Academy Working Conference on Practice-Driven Research on Enterprise Transformation, PRET 2009, held at CAiSE 2009, Amsterdam, The Netherlands, June 11, 2009, Proceedings*, Springer.

PUNTAMBEKAR, A. A. 2009. *Software Engineering*, Technical Publications.

PUNTAMBEKAR, I. A. D. A. A. 2008. *Systems Programming*, Technical Publications.

QIAN, K., FU, X., TAO, L. and XU, C. 2009. *Software Architecture and Design Illuminated*, Jones and Bartlett Learning.

RAYNARD, B. 2008. *Togaf the Open Group Architecture Framework 100 Success Secrets - 100 Most Asked Questions: The Missing Togaf Guide on How to Achieve and Then Sustain Superior Enterprise Architecture Execution*, Emereo Pty Limited.

REIS, H. T. and JUDD, C. M. 2000. *Handbook of Research Methods in Social and Personality Psychology*, Cambridge University Press.

REMENYI, D. 2006. *Proceedings of the 13th European Conference on Information Technology Evaluation*, Academic Conferences Limited.

REMENYI, D. 2011. *Field Methods for Academic Research: Interviews, Focus Groups and Questionnaires*, Academic Publishsing.

RITTGEN, P. 2007. *Enterprise Modeling And Computing With Uml*, Idea Group Pub.

ROSS, J. W., WEILL, P. and ROBERTSON, D. 2006. *Enterprise architecture as strategy: creating a foundation for business execution*, Harvard Business School Press.

RUNESON, P., HOST, M., RAINER, A. and REGNELL, B. 2012. *Case Study Research in Software Engineering: Guidelines and Examples*, Wiley.

SAHA, P. 2007. *Handbook of Enterprise Systems Architecture in Practice*, IGI Global.
SAHA, P. 2009. *Advances in Government Enterprise Architecture*, Information Science Reference.
SAHA, S. P., DOUCET, J. G. G., BERNARD, B. S. and BERNARD, S. 2009. *Coherency Management: Architecting the Enterprise for Alignment, Agility and Assurance*, AuthorHouse.
SALEH, K. A. 2009. *Software Engineering*, J. Ross Pub.
SALKIND, N. J. 2010. *Encyclopedia of Research Design*, SAGE Publications.
SANGEETA, S. 2008. *Software Engineering*, New Age International (P) Ltd.
SATZINGER, J. W., JACKSON, R. B. and BURD, S. D. 2008. *Systems Analysis and Design in a Changing World*, Course Technology.
SAUNDERS, M., LEWIS, P. and THORNHILL, A. 2009. *Research Methods for Business Students*, Pearson Education.
SCHOLL, H. J. 2010. *E-Government: Information, Technology, and Transformation*, M E Sharpe Incorporated.
SCHWALBE, K. 2010. *Information Technology Project Management*, Course Technology.
SESSIONS, R. 2007. A Comparison of the Top Four Enterprise-Architecture Methodologies.
SHELLY, G. B., CASHMAN, T. J. and ROSENBLATT, H. J. 2009. *Systems Analysis and Design*, Thomson Course Technology.
SIMANT, M. 2009. *Mechanical System Design*, Prentice-Hall Of India Pvt. Limited.
SIMONS, H. 2009. *Case Study Research in Practice*, SAGE Publications.
SMART, J. C. 2009. *Higher Education: Handbook of Theory and Research*, Springer.
SOBH, T. and ELLEITHY, K. 2010. *Innovations in Computing Sciences and Software Engineering*, Springer.
SPINDLER, G. and HAMMOND, L. 2006. *Innovations in Educational Ethnography: Theories, Methods, and Results*, Taylor and Francis.
STAIR, R. M. and REYNOLDS, G. W. 2011. *Fundamentals of Information Systems*, Course Technology.
SUOMI, R., CABRAL, R., HAMPE, J. F., HEIKKILÄ, A., JÄRVELÄINEN, J. and KOSKIVAARA, E. 2006. *Project E-Society: Building Bricks: 6th IFIP Conference on e-Commerce, e-Business and e-Government (I3E 2006), October 11-13, 2006, Turku, Finland*, Springer.
TAMBOURIS, E., MACINTOSH, A. and DE BRUIJN, H. 2011. *Electronic Participation: Third IFIP WG 8.5 International Conference, ePart 2011, Delft, The Netherlands, August 29 – September 1, 2011. Proceedings*, Springer.
TAYLOR, A. and PARISH, J. R. 2009. *Career Opportunities in the Internet, Video Games, and Multimedia*, Facts On File, Incorporated.
TAYLOR, E. A. 2006. *Research Methodology: A Guide For Researchers In Management And Social Sciences*, Prentice-Hall Of India Pvt. Ltd.
THE-OPEN-GROUP 2006. *Togaf 2007 Edition (Incorporating 8.1.1)* Van Haren Pub.
THE-OPEN-GROUP 2007. TOGAF version 8 Enterprise Edition.
THE-OPEN-GROUP 2009. *TOGAF Version 9*, Van Haren Publishing.
TOMKOWICZ, T. 2007. *Modeling as a mechanism to align Business Process Management with Enterprise Architecture: an analysis from the Financial Services Industry*, GRIN Verlag.
TROCHIM, W. M. K. 2006. *Deduction and Induction* [Online]. Available: http://www.socialresearchmethods.net/kb/dedind.php.
TUPPER, C. 2011. *Data Architecture: From Zen to Reality*, Elsevier Science and Technology.
UNHELKAR, B. 2010. *Handbook of Research on Green Ict: Technology, Business, and Social Perspectives*, IGI Global.
VAN, J. W. 2006. *Integrating Business Rules of Information Systems with Enterprise Architecture*, Lawrence Technological University.

VAN SANTE, T. and VAN DEN BENT, H. 2007. *Togaf the Open Group Architectural Framework: A Management Guide*, Van Haren Publishing.

VARAJAO, J. E. Q., CRUZ-CUNHA, M. M., PUTNIK, G. D. and TRIGO, A. 2010. *ENTERprise Information Systems, Part II: International Conference, CENTERIS 2010, Viana do Castelo, Portugal, October 20-22, 2010, Proceedings*, Springer.

VASUDEVA, V. 2009. *Software Architecture: A Case Based Approach*, Pearson Education.

VELLANI, K. H. 2007. *Strategic Security Management: A Risk Assessment Guide for Decision Makers*, Butterworth-Heinemann.

VIDGEN, R. and WOOD, B. 2002. *Developing Web Information Systems: From Strategy to Implementation*, Butterworth-Heinemann.

WAKEFIELD, S. and THIND, S. System Development Life Cycle(SDLC) andProject Risk Management. ISACA Fall Conference, Sept. 25-27, 2006 San Francisco, CA.

WATT, D. A. 2004. *Programming Language Design Concepts*, Wiley.

WESTFALL, L. 2008. *The Certified Software Quality Engineer Handbook*, ASQ Quality Press.

WIEBE, E., DUREPOS, G. and MILLS, A. J. 2009. *Encyclopedia of Case Study Research*, SAGE Publications.

WOLCOTT, H. F. 1999. *Ethnography: A Way of Seeing*, AltaMira Press.

YATES, S. J. 2003. *Doing Social Science Research*, SAGE Publications.

YIN, R. K. 2009. *Case study research: design and methods*, Sage Publications.

ZACHMAN, J. A. 1987. A framework for information systems architecture. *IBM System's Journal,* 26**,** 276.

ZAK, D. 2010. *An Introduction to Programming With C++*, Course Technology.

ZIEMANN, J. 2011. *Architecture of Interoperable Information Systems: An Enterprise Model-Based Approach for Describing and Enacting Collaborative Business Processes*, Paul Elek Incorporated.